

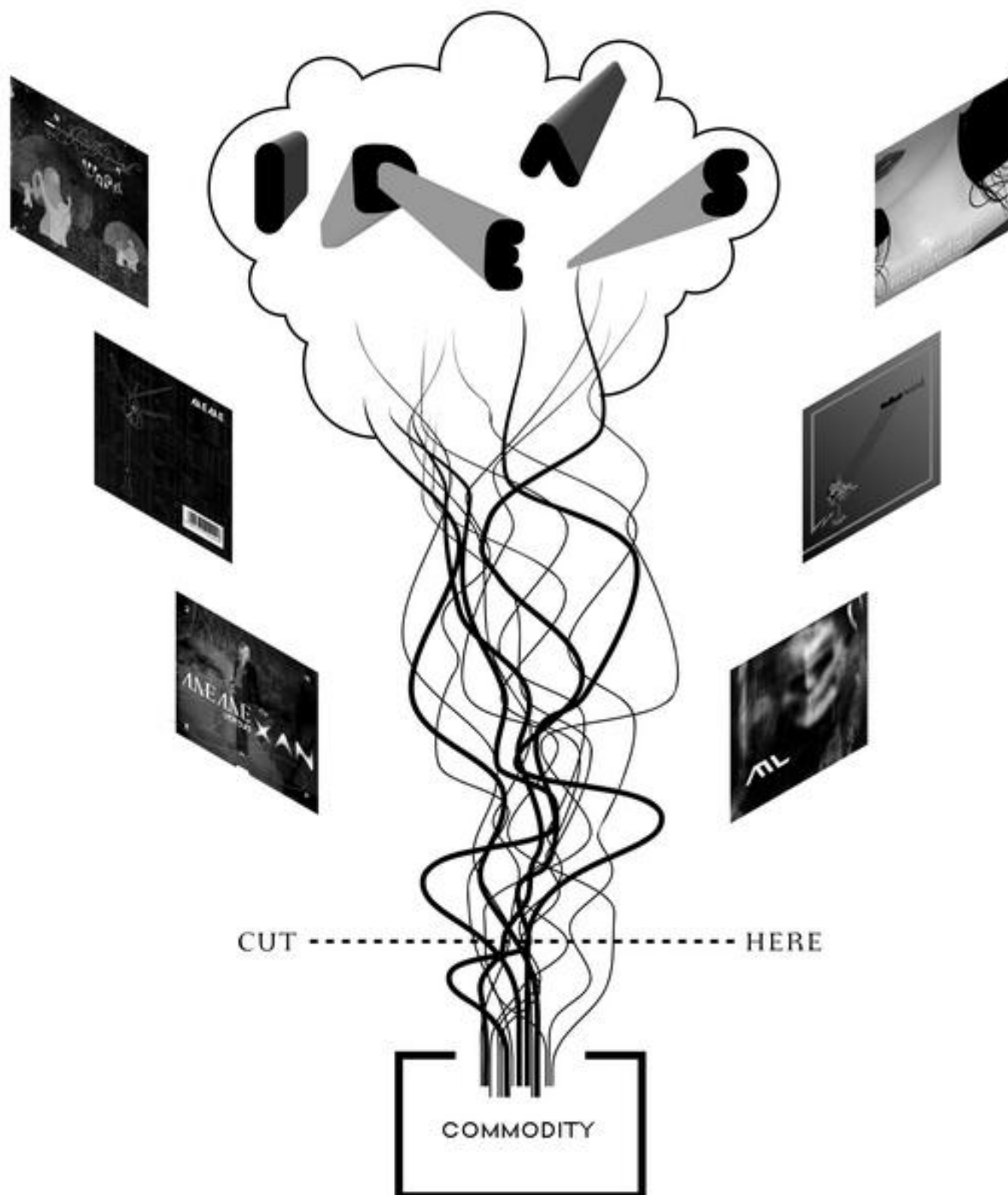


LOCARECORDS.COM

PIRATE
THIS MUSIC

Ideas are not objects that can be wholly owned - they can be infinitely copied and re-used due to their immaterial form.

All are music and the accompanying artwork is released under Creative Commons Attribution Share Alike Licence



We are always interested in hearing from creative people - please contact us at info@locarecords



**Communicating
mathematics...**



**...is getting
easier.**



Math Focus
News
Discussion
Collaboration
Software
Books

**Powerful tools
for online
mathematics**





Contents

Issue 1, February 2005

EDITORIAL

Welcome to the first Free Software Magazine 7

Mac OS X: Welcome to the jungle 30

by Chris J. Karr

A look inside the Mac OS X software ecology

FOCUS

Format Wars 8

by Marco Fioretti

File formats: the past, the present and a possible future

The magic of live CDs 35

by Harish Pillay

What are live CDs, and how do they work?

XML: the answer to everything? 12

by Kay Ethier, Scott Abel

This article weighs the pros and cons of XML for some applications (publishing), and explores why it is the best possible solution for many programming and publishing needs.

Every engineer's checklist for justifying free software 39

by Malcolm D. Spence

Free software is not just about "no license fees"!

Free file formats and the future of intellectual freedom 17

by Terry Hancock

Information as property may be served by closed file formats, but the freedom of information requires free formats

Smarter password management 45

by John Locke

How to handle your passwords without getting lost

TECH WORLD

Creating Free Software Magazine 24

by Tony Mobily

A long path that takes us to the very beginning of this project

The content tail wags the IT dog 49

by Daniel James

Without hardware and software, there would be nothing for digital media to be created on, or used with. And yet the content industry attempts to tell the far larger IT industry what it can and cannot do.

WORD WORLD

Motivation and value of free re- sources 52

by Aaron E. Klemm

Wikipedia and PlanetMath show the way

It's all about freedom 57

by Christian Einfeldt

Freedom is free software's competitive advantage

The Commons 61

by David M. Berry

The Commons as an Idea - Ideas as a Commons

Let's not forget our roots 66

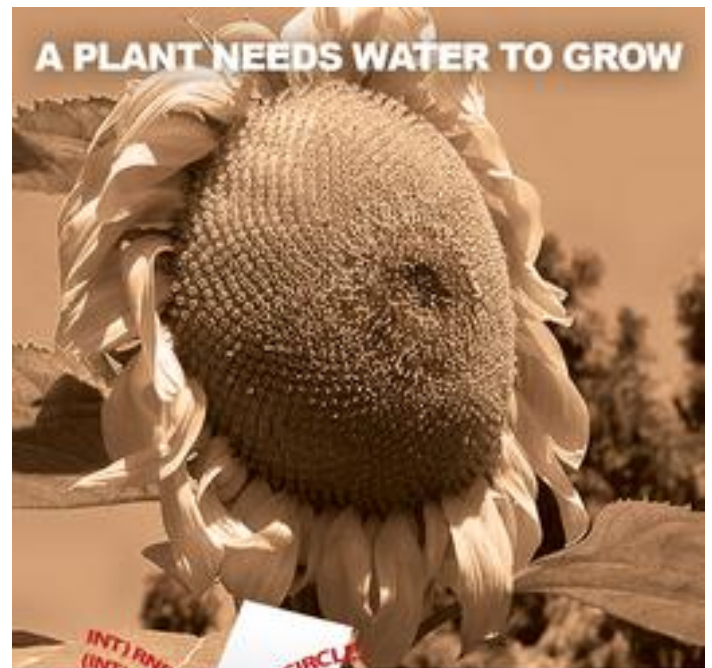
by Tom Chance

Free software is not just about cost or stability: free software is a movement that mustn't forget the principles which made it possible

Richard Stallman's blog 69

by Richard Stallman

Richard's [blog](http://agia.fsf.org/rms-blog) (<http://agia.fsf.org/rms-blog>), from September 2004 to October 2004



OpenSource Labs

Pioneers in Open Source Solutions



"Open Source Labs, a division of Vyom Labs, is a centre of excellence providing world-class support, consultancy and training across the entire spectrum of open source technologies. Open Source Labs enables organizations worldwide to gain maximum value from Open Source Software while reducing the total cost of ownership."

Open Source Support Services

We provide migration, installation, and post installation support and technical support solutions for most Open Source technologies :

- Linux and MySQL,
- Software Engineering tools like CVS, BugZilla
- Software IT management tools like Nagios, MRTG, OTRS
- Messaging solutions including Qmail, Sendmail, WebMail, SpamAssassin, Thunderbird
- Custom Open Source development in PHP, Perl (using MySQL or Postgres SQL servers)

Linux Training

- Linux OS Administration
- Networking and Security
- RedHat Linux
- SuSe Linux

Developer Course on Linux

- OSL 101 Unix Shell scripting
- OSL 102 PHP/Perl + MySQL



Authorized Red Hat Training Partner



OpenSource Labs, (A division of Vyom Labs Pvt Ltd), Dayaprabha House, Opp Sulzer India
ITI Road, Aundh, Pune 411007 **INDIA**. Ph: +91 982278700, 9120-25889236
e-mail : contactus@opensource-labs.com

www.opensource-labs.com

Welcome to the first Free Software Magazine

Iwould have liked to start this editorial defining what free software is, but I found myself writing – and deleting – my sentences time and again. The problem is that free software means different things to different people. To some, free software is a way to save money in licensing fees and technical support. To some, it's a way of sharing their skills (which they do for different reasons: research, personal development, money, etc). And to others free software is a movement, a way of life.

Whatever the case, due to its many merits, free software's popularity is growing daily. Even non-geeks are discovering that most of the web sites that they visit run on free software (Apache); there is a valid alternative to Internet Explorer (Firefox); and their internet provider's network is secured by free software (Nexus, free firewall, etc).

And yet, until today there hasn't been a single magazine dedicated entirely to free software.

As the Editor in Chief, I'm very excited because I've always wanted to be involved in a project like this. I've always considered myself a "free software consultant and advocate", but I have never felt that I was giving enough back to the community. In a way, I consider Free Software Magazine to be my big opportunity – and I believe that it's a big opportunity for free software, its users and its programmers. All of the articles are released under a free license six weeks after publication. This means that we'll steadily build up a library of valuable material, which can then be used both in technical and non-technical discussions by the public at large.

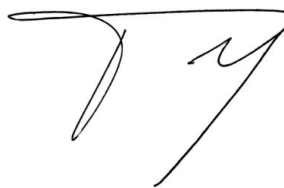
Now, this project is not risk-free. In the publishing industry you need *numbers* to make everything work. The more you print, the less you pay. The more readers you have, the more likely you are to get paying advertisers and so it goes on. At the moment, nobody really knows what these numbers will be for a magazine on free software, simply because there's never been one.

I believe that we (myself, the staff, and the contributors) did a fantastic job, and it shows. If you don't think we did, and you believe that Free Software Magazine isn't up to standard, please let us know - we welcome any criticism.

If you believe in this project, please let the whole world know about it, use all those means that made great free software projects successful: talk about Free Software Magazine in your blog, user group mailing lists, social networks, professional web sites, IRC, etc. This way, you will help the magazine gain momentum and obtain the exposure it – and free software – deserve.

I'll see you here next month!

Copyright information © 2005 by Tony Mobily
Verbatim copying and distribution of this entire article is permitted in any medium without royalty provided this notice is preserved.



Free Software Magazine is a magazine by The Open Company Partners Inc, 90 Main St. Road Town, Tortola BVI

EDITOR IN CHIEF

Tony Mobily

TECHNICAL EDITORS

Clare James
Pancrazio De Mauro
Gianluca Insolubile

EDITORS

Anna Dymitr Hawkes
Dave Guard

TECHS

Gianluca Pignalberi (L^AT_EX class and magazine generation)
Gian Maria Ricci (RTF to XML converter using VBA)

GRAPHIC DESIGN

Alan Sprecacener (Web, cover and advertising design)
Tony Mobily, Gianluca Pignalberi, Alan Sprecacener (Magazine design)

THIS PROJECT EXISTS THANKS TO

Donald E. Knuth, Leslie Lamport, People at T_EX Users Group **TUG** (<http://www.tug.org>)

For copyright information about the contents of Free Software Magazine, please see the section "Copyright information" at the end of each article. If an article is released under a free license six weeks after publication, and the six weeks are not over yet, you may not reproduce or retransmit the article, in whole or in part, in any manner, without the prior written consent of the author.

Format Wars

File formats: the past, the present and a possible future

Marco Fioretti

Real programmers love their applications' source code: the faster and more elegant it is, the better. Users are after very different things: they seem to want simplicity, flashy colors, nice icons and tons of options. In spite of these reasons, or perhaps because of them, programmers and users often forget what lies in the middle of it all: information.

Who owns the information?

Almost all software applications are used to manage *information* so these applications are worthless without information to process, store and display. For example, you could use a word processor to write letters or video editing suites to edit footage of your girlfriend at the beach.

Almost all software applications are used to manage information so these applications are worthless without information to process, store and display

If information exists before (and independent of) the applications, the file format used to store the information should be defined before hand. In this ideal situation, you could potentially write several programs (released under free or non-free licenses) to handle your information.

Please keep in mind that here "information" means *any* kind of creative work: blog entries, private movies, essays, government reports, court rulings, road projects... In an ideal world, the format used to store this information doesn't matter: it should simply belong only to its author, or whoever paid for its production.

Fig. 1: An OpenOffice RTF file opened with Word X for Macintosh

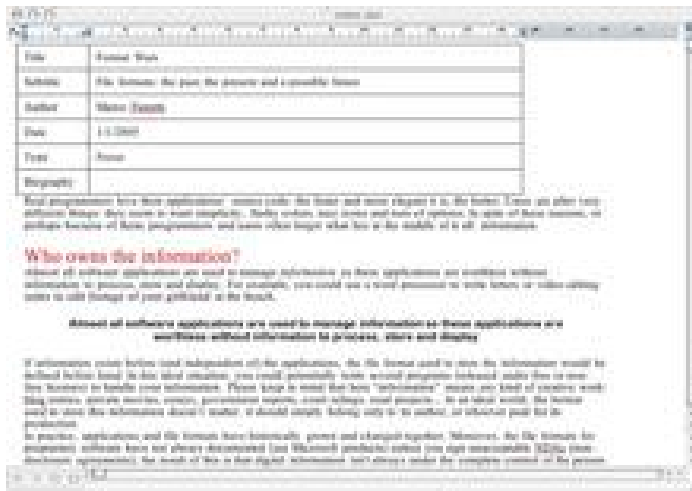


In practice, applications and file formats have historically grown and changed together. Moreover, the file formats for proprietary software have not always been documented (see Microsoft products) unless you sign unacceptable NDAs (Non-Disclosure Agreements); the result of this is that digital information isn't always under the complete control of the person who created it.

In my opinion this problem has been underestimated for a long time, probably because in the beginning people didn't think it was such a big deal.

First of all, far fewer people had computers. When they did have them, they weren't often networked and were physically incompatible (think of Mac and PCs, which even had problems sharing a floppy disk!). Resources were very limited: monitors, processors and hard drives weren't even remotely comparable to what we have today, and therefore visually "fancy" information wasn't as important as it is today (think WYSIWYG). Even complex spreadsheets were

Fig. 2: The same OpenOffice RTF file opened with Word 2004 for Macintosh



stored as CSV format (plain text separated by commas) or as binary files. Back then, there was a situation similar to today's: if the information was stored as text files, you could use powerful text processing tools like sed, awk and then Perl. If it was stored in binary format, reverse engineering and black magic fixed most of the problems. Exchanging information at that point wasn't often a problem; even when binary-only format became more common thanks to WordStar and AutoCAD, the end product was nearly always a stack of paper that was to be shipped or archived somewhere.

This paper could then be read even centuries after it was written, without a concern for what "brand" of paper, or which printer or pen had been used to write on it.

In a way, paper was the *lingua franca*.

In a way, paper was the *lingua franca*

Today, with the internet, CDs and search engines, any file can be used and distributed in several different ways without ever turning into durable, non proprietary (and non-searchable, I must add), printed paper. Talk about progress...

Today's scenario

Today's scenario is somehow very similar to what it was a few years ago - just a bit more complicated. Proprietary file formats are now more complex than before and therefore harder to reverse-engineer. Text-based file formats are still based on text (obviously!), but they have gained a level of

complexity as well: rather than representing the information directly (like plain text documents or CSV spreadsheets do), they are usually based on XML.

For example, the content of a cell in OpenOffice.org could be represented with this:

```
<style:properties style:column-width="1.785cm"/>
\ldots{}
<table:table-cell><text:p>600000</text:p>
</table:table-cell>
```

These two lines above simply state that the width of the column containing this cell must be 1.785 cm and that the cell stores the number 600000.

A paragraph in a letter could be:

```
<p>This is the <b>first</b> paragraph</p>
<p>This is the second one</p>
```

The advantages of XML files are clear: anybody can write an application which manipulates them, as long as they know what every XML tag means in that specific context.

A word on encoding

Even "plain text" can mean different things, depending on how it's *encoded*. The encoding defines which sequence of bits represents a particular character (such as a letter, a white space, symbols like "©" and "#", and so on) used in a written language.

In ASCII (*American Standard Code for Information Interchange*), for example, the sequence "01000001" corresponds with the capital letter "A".

Even "plain text" can mean different things, depending on how it's *encoded*

The ASCII encoding (or format) is really ubiquitous these days, but has simply outlived its meaning in a wired world where most people *don't* speak English. Over the last few years many more types of encoding have been created in order to deal with almost any other language on the planet including non-alphabetic ones (Chinese, Hindu, Korean, Japanese...). The resulting confusion has been made worse by the fact that "plain text files" don't contain, by definition, any headers to declare their internal encoding. Consequently, the programs processing them have to guess, or be told, which encoding they should use to display them; otherwise, blank or strange characters are displayed instead of the correct ones.

Today, the Unicode family of standards provides a definitive solution; unfortunately, it will take a lot of effort and time to have it accepted - not to mention used - everywhere. When, in 2002, Red Hat Linux switched to Unicode, many people complained on mailing lists because “everything had become slower, just to make the French happy”.

The problem with closed file formats

Why should end users care at all about these issues? Because in the last two decades at least, file formats have been used to avoid free market competition, making it harder for customers to switch to newer and better products, or to place restrictions on how people use programs or the information produced with them.

In the last two decades at least, file formats have been used to avoid free market competition, making it harder for customers to switch to newer and better products

This is evident in fields as varied as office automation, industrial design and video streaming. In the first case, almost every user knows that the only guaranteed way to open “.doc” or “.xls” files *reliably* is by using the same version of Microsoft Word or Excel which created them in the first place. Remember that this applies to *all* of your files, starting from your personal diary...

When it comes to engineering, many projects for buildings, mechanical parts, furniture and bridges are stored in the DWG file format of AutoCAD, produced by Autodesk. In 1998, competitors launched cheaper products based on an equivalent format. Autodesk's reaction was not limited to improving features, service and discounts. Their advertising campaign focused on reminding people that only Autodesk's products were 100% capable of keeping *existing* projects completely accessible. The full story can be read online in the FAQ and history pages on the website of the **Open Design Alliance** (<http://www.opendesign.com>) founded just to create an alternative file format.

What about multimedia? MPEG-4 is an advanced format for compressed video: DivX and many other decoders are based on it. Now, according to the **MPEG-4 License page** (www.mpegla.com/m4v/m4v-faq.cfm):

Fig. 3: The home page of the Open Design Alliance



Video providers who receive remuneration for offering MPEG-4 video either directly (e.g. subscription or title-by-title fees) or indirectly (e.g. advertising or underwriting fees) pay a royalty for the right to use the decoders and encoders to receive and transmit the remunerated video.

The fear of having to pay MPEG-4 fees even when you place banners and video clips of your holidays on your home page has been enough to start projects like **Theora** (<http://www.theora.org>).

XML: the savior?

The cases above are just a few examples of how file formats have effectively been used to enforce a much greater control on end users than was possible before.

The family of technologies known as XML (*eXtensible Markup Language*) can play an essential role in solving these problems (at least in some areas).

XML was designed to make it easy to *exchange* information rather than locking it.

XML was designed to make it easy to exchange information not easy to lock information

XML files are in a plain (Unicode!) text format similar to HTML. This alone makes reverse-engineering of XML files much easier, compared with binary formats. Of course text can never be as compact and fast to parse as pure binary data, but it has a huge advantage: it can be processed with

Fig. 4: The OASIS consortium's home page



any of the existing text-processing tools, known to and improved on by Unix users since the '70s.

For example to extract the XML code shown above I only had to unzip the original spreadsheet and open the *content.xml* file with a text editor.

However, XML is no more or less proprietary or open than binary formats. Its full benefits are only available when it's completely and openly documented, guaranteed to stay that way and, above all, legally usable without asking permission or paying fees to anybody.

Format wars: the next episode

The **OASIS consortium** (<http://www.oasis-open.org>) produces open XML standards in all fields of business and computing activity. Perhaps its most important achievement is the OpenDocument format for word processing, spreadsheets and presentations, directly derived from the one used in OpenOffice.org and submitted to the International Standard Organization (ISO).

OpenDocument is more powerful than XHTML and, unlike other formats, there are already some cross platform applications which use it. OpenOffice.org 2.0, due for release around March 2005, will use it by default, and other products, from **Koffice** (<http://koffice.kde.org/>) to IBM's Workplace and servers like **Plone** (<http://www.plone.org/>), can already read and write files in this format. Just add a Firefox plug-in, and OpenDocument will be immediately accessible from your browser! For these reasons, some people think that it could eventually replace HTML as the default format for the internet.

However, there is no need to look that far. There is something much more important already happening today. Namely, the European Union (EU) wants to make it possi-

ble for all EU public administrations to (re)take ownership of the documents they manage on behalf of their citizens. In order for this to happen, these administrations, or anybody willing to do business with them, will eventually have to produce, exchange and store files in the right format.

In 2003 an EU study called the **Valoris Report** (<http://europa.eu.int/ida/en/document/3439>)

concluded that an XML file format, highly portable and very open, is required to reach this goal. The report mentions the efforts in this field by Sun (OpenOffice.org/OASIS) and Microsoft (MSXML), pointing out several limitations of the latter. The main limit is the fact that Microsoft prefers not to completely separate the file format from the applications. They would much rather assist *selected* partners in enabling their applications to read and interoperate with MSXML. It doesn't sound like much of a concession, does it?

This episode of the Format Wars is still being quietly fought while we're writing (mid December 2004): stay tuned for further news. Hopefully, if it all ends as it should, the utopia described at the beginning of the article, can come into being: that file formats are defined *before and independently* of any implementation, in any field of computing.

Conclusions

In my opinion, one of the best signs that software is still in its infancy is the way this issue of formats has been ignored so far, by professionals and casual users alike. Luckily the tide has started to turn. Things like *OpenDocument* are certainly steps in the right direction. Nobody can predict what combinations of proprietary and free software will be used twenty years from now. The most probable guess is that there will be a lot of them, and each user will be free to choose the best combination for his or her real needs. In any case, I hope that in twenty years the era where information is locked up by proprietary and application specific formats will be just a laughable memory.

Copyright information

© 2005 by Marco Fioretti

Verbatim copying and distribution of this entire article is permitted in any medium without royalty provided this notice is preserved.

About the author

Marco Fioretti is a freelance writer based in Italy

XML: the answer to everything?

This article weighs the pros and cons of XML for some applications (publishing), and explores why it is the best possible solution for many programming and publishing needs.

Kay Ethier

Scott Abel



his article weighs the pros and cons of XML for some applications (publishing), and explores why it is the best possible solution for many programming and publishing needs.

Everywhere you turn these days, someone is talking Extensible Markup Language (XML). Jump into a discussion about publishing - XML is touted as a means of exchanging information. Talk with someone about the new software tool she is creating - she describes setting up some of her actions in XML. Ask a webmaster what he's been doing - he raves about the dynamic content he's serving up to site visitors using XML from a database. In short, XML is a great solution to a wide variety of challenges, and it seems to be everywhere. But is it the cure for every data or content challenge? The simple answer is, no.

Everywhere you turn these days,
someone is talking Extensible Markup
Language (XML)

Not everyone needs XML to make things work. For some small organizations, publishing processes are straightforward enough that the costs of implementing an XML solution may not be worthwhile. But the only way to be sure is to perform a thorough examination of the business processes and review cycles that produce information products - most organizations and companies grossly underestimate the amount of information they could potentially reuse in publishing, and overestimate the costs of reusing that information with an XML-based solution. And they're not aware of the breadth of available free tools that can get them well on the road to their XML destination.

What is XML?

XML is meta markup language that is used to create new markup languages. It's most commonly used to create tag sets and processing instructions that describe structured content for presentation in text documents, but it can also be used to describe, manage, and deliver content of all types (text, images, voice, forms, multimedia files, and so on) and to transform transactional data between disparate database systems.

XML is meta markup language that is
used to create new markup languages.

Unlike Hypertext Markup Language (HTML), which is a display markup language with a predefined list of tag sets designed solely to control how information is presented in a web browser, XML presents content in an open, standards-based, media-neutral, operating system-agnostic, platform-independent format. XML is extensible because it allows organizations to define their own sets of tags, each with a meaningful (semantic) "name". Semantic names (or tags) are more useful than generic HTML tags because they can describe content in real-world, user-friendly and context-specific ways. For instance, the XML tag `<product name>` is much more descriptive than the HTML tag `<h2>`.

In a traditional word processing environment, the formatting data is stored with the content it governs, and changes to the formatting involve changes to the content itself. XML's strength comes in its ability to separate content from formatting data, thus allowing authors to create content with-

out spending unnecessary time formatting that information. XML style sheets control the formatting of the content being created, and specify how it will be presented in each medium.

XML content can therefore be automatically transformed (with the help of style sheets) from a single text source into a variety of information products (printed product brochures, web site content, wireless content, etc.) each with its own look and feel. And, XML content can be personalized and delivered dynamically on the fly, based on the specific requirements of the end user.

XML also differs from HTML in that it allows documentation to be processed by computer software programs, thus allowing organizations to reuse content from disparate data repositories, and recombine that data in ways – and in various media - not possible with HTML. XML supports single source content reuse, and allows organizations to make changes to a content element (like a product description) and have those changes reflected instantly and automatically in every information product that uses that information, regardless of the medium. This ability to reuse information and to make changes once and have them appear globally saves organizations considerable time and money revising, updating, and translating content.

XML content is also “validated” against document guidelines encoded in a Document Type Definition (DTD) and can enforce standards on the authors who develop content. This ability is particularly useful in validated or regulated environments (life sciences companies, legal firms, automobile and aerospace industries, the financial sector) in which completeness, consistent structure, and accuracy of information are all essential, if costly regulatory compliance and legal issues are to be avoided.

W3C Goals for XML

After the world wide web explosion, web users were inundated with miles of good and bad HTML, and the W3C sought a better solution for publishing, cataloguing, locating, retrieving and archiving data. The guidelines they set for this “something better than HTML” resulted in the development of XML. The “design goals” for XML, which set it aside from HTML, include the following (source: [W3C \(http://www.w3c.com\)](http://www.w3c.com)).

1. XML shall be straightforwardly usable over the Internet.
2. XML shall support a wide variety of applications.
3. XML shall be compatible with SGML.

4. It shall be easy to write programs, which process XML documents.
5. The number of optional features in XML is to be kept to the absolute minimum, ideally zero.
6. XML documents should be human-legible and reasonably clear.
7. The XML design should be prepared quickly.
8. The design of XML shall be formal and concise.
9. XML documents shall be easy to create.
10. Terseness in XML markup is of minimal importance.

This article focuses primarily on the second W3C requirement for XML, that it plays well with a variety of tools that perform various tasks. And since the potential uses of XML are countless, and space is limited, we’ve restricted our scope to the use of XML in publishing.

XML Uses

In the publishing arena, XML is used by authoring and content management tools. Authors use the XML elements and attributes to produce documents. Content management tools use the XML elements and attributes as data that can be retrieved or marked for reuse.

Is this the answer to everything? Well, in the publishing world the answer is sometimes “no”, because affordable publishing can sometimes be accomplished without the help of XML - XML would be overkill. However, XML often is the best option for organizations that take the time to evaluate their content lifecycle and to examine how much it costs to create, maintain, translate, deliver, store, reuse, archive, and retire content. A recent study by ZapThink (“XML in the Content Lifecycle Foundation Report Creating, Managing, Publishing, Syndicating, and Protecting Content with XML”) found that the biggest - and most expensive - challenge for most organizations today is content reuse. The study found that “Producers of content in the enterprise spend over 60% of their time locating, formatting, and structuring content and just 40% of their time actually creating it.” (Source: [ZapThink \(http://www.zapthink.com/report.html?id=ZTR-CL100\)](http://www.zapthink.com/report.html?id=ZTR-CL100))

The sad fact is, most organizations don’t know how much their content creation and management efforts cost them, and so they assume that XML is not for them. The reality is that the only way to know whether XML is the right choice for your organization’s publishing needs is to seek the assistance of a content management expert who can perform an organizational needs analysis, a content lifecycle analysis, and an audit of your existing content. Additional

services offered by content management consultants include customer needs analysis, tools recommendations and assistance calculating return on investment. Analysis often identifies obstacles to change (tools, processes, and people) that will need to be addressed before you adopt XML as a publishing solution. Once you know how much it costs, and what obstacles you'll face, you can make an informed business decision about whether to move to XML publishing or not.

XML does provide a lot of options. Exchanging content, for example, is often easier and more affordable with XML than it is with proprietary tools like Microsoft Word. Rather than saving content in a proprietary format, authors can output their document content into XML and pass it along to colleagues or customers who need the content but who may use other authoring and publishing tools. Additionally, XML makes reuse of information easier since formatting data is separated from XML content. Separating content from format is one of the biggest productivity gains an organization can obtain by adopting XML.

Exchanging content, for example, is often easier and more affordable with XML than it is with proprietary tools like Microsoft Word

XML content may be used to produce one document, and that same XML content can then be harnessed to create additional documents, each with a completely different look and feel. Alternatively, the same XML content can be dynamically served up to various audiences in different chunks or in different sequences using other technologies (see "XSLT", below). This represents a degree of flexibility that HTML simply doesn't offer.

Free XML Authoring Tools

There are a wide variety of free XML authoring tools available for download on the internet. Each has its own strengths and weaknesses, and no one free tool does it all (i.e. your mileage may vary).

Check them out and learn as much as you can about XML authoring before you decide to employ any particular tool:

- **Altova Authentic** (http://www.altova.com/products_doc.html)
- **XML Cooktop** (<http://www.xmlcooktop.com/>)

- **Open XML Editor** (<http://www.philo.de/xmledit/>)
- **Xray2** (<http://architag.com/xray/>)

XML-related Technologies

Jonathan Robie (<http://www.gca.org/papers/xmleurope2001/papers/bio/s13-1auth2.html>), an XML Research Specialist at Software AG, once

exclaimed, "XML doesn't do anything!" In its purest sense, this is true; by itself, XML will not magically repurpose content for multiple media or audiences. XML doesn't provide formatting in the absence of additional technologies. In order to make XML "look good", or turn it into a final deliverable, some assistance from format-conscious technologies is required... but on the other hand, no amount of such formatting technology can turn ugly-duckling HTML content into a coterie of media swans.

XSL and XSLT

In the HTML world, Cascading Style Sheets ("CSS" files) make HTML display as desired... in a web browser. Because XML separates content from its formatting data, you must employ additional technologies to format XML, allowing it to display as you wish. XML can be formatted a few different ways. You can bring XML content into XML-based tools to change its appearance. (You can also use HTML to format XML.) The XML formatting and transforming language (Extensible Stylesheet Language, Transform, "XSLT" for short) can adjust XML output for various display purposes. When you have multiple media in which you want to present your content, XML is far more flexible than its HTML ancestors.

XSLT uses the tags within an XML document to control formatted output. Formatting XML content can be as simple as adding bold to a <companyname> tagged object. The formatting can be as complex as telling all of the pieces of an invoice, for example, to display in a certain font, point size, style, etc. in a table and make the table content "sortable" by any of the tags used in your XML content.

Free software tools used for XSLT include Saxon and Xalan (and others). Each allows you to perform transforms without moving your XML content into a proprietary tool that will "trap" you into using that tool in future.

Saxon, created by Michael Kay, is available in several flavors. The "lite" version allows you to do transformations on any PC running the Java Runtime Environment (JRE).

Saxon is available via [Michael Kay's SourceForge web-site](http://saxon.sourceforge.net/) (<http://saxon.sourceforge.net/>) The JRE is available from multiple sites, including [java.com](http://www.java.com/en/download/windows_automatic.jsp) (http://www.java.com/en/download/windows_automatic.jsp).

Xalan is an XSLT processor designed to transform XML documents into HTML, text, or other XML document types and is available via [The Apache XML Project](http://xml.apache.org/xalan-j/) (<http://xml.apache.org/xalan-j/>) among other sites.

Free software tools used for XSLT include Saxon and Xalan (and others). Each allows you to perform transforms without moving your XML content into a proprietary tool that will "trap" you into using that tool in future

A good resource for more information on working with XSLT and XML is Mitch Amiano's free software collection, the "Agile Markup Toolkit", which is available at no cost. The CD itself contains several dozen free software installations and links. Any software on the CD also includes reference information that indicates where it came from, allowing you to update as new releases become available. Mitch is a big user of free software, very involved in the free software community, and is also a user of the tools he has gathered on this CD.

Visit the [Agile Markup Toolkit's web site](http://home.agilemarkup.com/index.php?option=content&task=view&id=55&Itemid=29) (<http://home.agilemarkup.com/index.php?option=content&task=view&id=55&Itemid=29>) for more information about "Agile Markup Toolkit".

XSL-FO

Another subset of XSL is XSL-FO. The FO stands for "formatting objects." XSL-FO provides a means for formatting XML for presentation. More information on its capabilities is available at [the W3C website](http://www.w3.org/TR/xsl/) (<http://www.w3.org/TR/xsl/>).

XQuery

Some companies may be publishing information stored in a database or even stored as XML. XQuery allows you to query XML, similar to the way SQL is used to access databases. More information, and a

great overview, are available from [Data Direct Technologies](http://www.datadirect.com/techzone/xml/basics/basics/index.ssp) (<http://www.datadirect.com/techzone/xml/basics/basics/index.ssp>).

XML Performance

How has XML met with the [W3C expectations](http://www.w3.org) (<http://www.w3.org>)? Certainly there are many XML-driven websites. Check out Safari, CNN, Fidelity, and Wired, among others. These are dynamically generated pages with XML behind the scenes. At Fidelity, XML ties together web and back-end systems to deliver hundreds of thousands of transaction per hour to its web site customers. Fidelity says it's realizing millions of dollars of savings in infrastructure and development costs by eliminating the need for transformation of data between the company's disparate database systems and by reducing (by 50%) the number of web application servers through which customer data travels. (Source: [InternetWeek](http://www.internetweek.com/news/lead01/lead080601.htm) (<http://www.internetweek.com/news/lead01/lead080601.htm>)).

In publishing, XML has proven beneficial for creating materials derived from information stored in a database or publishing information that developers have created in XML. Some tools can open the XML and style it, providing paragraph formatting along with page layout (and in publishing, presentation is everything!). Such tools, which can automatically style XML, make publishing data easier and more affordable than traditional publishing methods.

However, XML can slow performance, if not integrated properly and appropriately planned for. "Research by IBM Labs shows that even small XML-based documents can increase the CPU cost of a relational database transaction by up to 10 times in the absence of a dedicated XML processing engine. The research concluded that XML parsing could have a 'potentially fatal impact' on high-performance, transaction-oriented database applications that use XML." (Source: [nwfusion.com](http://www.nwfusion.com/news/2004/0503xmlaccel.html) (<http://www.nwfusion.com/news/2004/0503xmlaccel.html>)). Hardware vendors are rushing to develop new gigabit-speed silicon to address the spread of XML and the processing problems it can sometime cause.

Again, it's important to employ a content management expert with experience in planning and implementing XML solutions before you adopt XML in your organization. XML is a business solution, not an IT solution. Employ it only after developing and conducting a thorough analysis of your organizational business needs, the needs of your

customers, and after evaluating your content lifecycle. The results should yield a unified strategy for XML use across your enterprise that will provide measurable benefits and a positive return on investment.

Conclusion

XML isn't the universal panacea... but it is often preferable to alternatives. Particularly in publishing applications, which represent so many ways data can be caught up in proprietary systems, it's a good idea to use non-proprietary technologies for content authoring, management and delivery, and it's crucial to assess and quantify the potential pay-backs of XML versus HTML systems.

Copyright information

© 2005 by Kay Ethier, Scott Abel

This article is made available under the "Attribution-NonCommercial-NoDerivs" Creative Commons License 2.0 available from <http://creativecommons.org/licenses/by-nc-nd/2.0/>.

About the author

Kay Ethier is an Adobe Certified Expert in FrameMaker 7.x and several prior versions. She instructs training classes, performs consulting, and provides support to clients in a variety of industries. Kay resides in the Research Triangle Park area of North Carolina and works for Bright Path Solutions (<http://www.travelthepath.com>). In 2001, Kay co-authored the book *XML Weekend Crash Course* (Wiley/HungryMinds). She has most recently been a contributing author on *Advanced FrameMaker* (TIPS Technical Publishing) and *XML and FrameMaker* (Apress).

Scott Abel is a technical writing specialist and content management strategist whose strengths lie in helping organizations improve the way they author, maintain, publish and archive their information assets. Scott has considerable experience in XML-based enterprise content management strategy, information architecture, software tool selection, high tech marketing communications, business process analysis, technical writing and editing, usability, content localization, and designing single source solutions.

A PLANT NEEDS WATER TO GROW

INT) RND) CIRC) BTNC) CIRC) (INT) RND) CIRC) BTNC) CIRC) (INT) RND) CIRC) BTNC) CIRC) (INT) RND) CIRC) BTNC) CIRC)

Free Software MAGAZINE

BY SUBSCRIBING YOU WILL BE SUPPORTING A MAGAZINE WHICH BELIEVES IN FREE SOFTWARE. ALL OUR ARTICLES ARE RELEASED UNDER THE GNU FREE DOCUMENTATION LICENSE, ENHANCING EXISTING INFORMATION ON FREE SOFTWARE.

SUBSCRIBE NOW!

WWW.FREESOFTWAREMAGAZINE.COM/SUBSCRIBE

THE OPEN VOICE NEEDS SUBSCRIBERS TO PROSPER!

Free file formats and the future of intellectual freedom

Information as property may be served by closed file formats, but the freedom of information requires free formats

Terry Hancock

So far, proprietary formats have been maintained through a number of short-term tricks, but the advantages of free formats become clearer in the long run. Business and the computer industry have tended to be very shortsighted. However there are some important classes of technically proficient users with a much longer outlook, whose needs can only be met by free file formats. If we in the free software community want to see free formats take hold, we need to address the needs of *these* users. We need to do this in order to leverage their interest into long-term acceptance of free standards by the world at large. We also need to ensure free standards *exist* — because in many key areas they just don't. Fortunately, the record provides good evidence that free software developers will step forward to meet these needs, once they become aware of them.

The need for ten to twenty years of data stability is routine in the sciences

My introduction to computers came as much through my interest in astronomy and spaceflight, as through any interest in computers for their own sake. One of the biggest differences is the perception of time — in the computer and IT world, three years may seem like an eternity, but there is still active research being done based on 30-year-old space technology (everything needed to get to the Moon), or even 300-year-old astronomical data (Galileo's drawings of the Great Red Spot on Jupiter). More mundanely, the need for ten to twenty years of data stability is routine in the sciences, and there is considerable likelihood that this need for stabil-

ity will increase in future, as these fields push the theoretical limits of detection and measurement accuracy.

From that perspective, any proprietary software's entire existence - let alone the duration of *any* commercial file format, or indeed the *concept* of file formats itself — is fleeting ephemera. To such users, the difficulties of closed file-formats are more likely to be blamed on the electronic medium itself, which is itself still regarded as a new development, even after 30–50 years of use.

As a researcher, I've also spent a lot of time using libraries, which as any serious researcher knows, are still far better than Google, simply because there's an awful lot of data that isn't available electronically, let alone for free and posted on the web. The search engines I first learned to use were the electronic library catalogs that became ubiquitous in the 1980s. Those systems ran on the MARC database standard that endures in modern library systems, although more modern standards like FRBR and Dublin Core are being developed.

To this day, there remains a public perception that libraries and the internet are somehow opposing forces in the world, with librarians clinging to worn-out paper technology in the face of the inevitable onslaught of better electronic methods. Maybe in 1980 that perception was true of many librarians, but in 2005 it's total bunk. Many, many librarians are excited about and are fully embracing the idea of *electronic libraries* — systems which combine the best of the web technologies with the tried-and-true methods that librarians have been using in their cataloging systems for decades, and in the processes of *document imaging* whereby they can convert existing print media to be remotely accessible. But they are encountering resistance, not just from the natural difficulties of the technology, but also from the artificial obsta-

cles created by copyright laws which have been made more restrictive than ever in the form of the Digital Millennium Copyright Act (DMCA) and made essentially immortal by the several copyright extension acts that have been passed since 1978. Finally, the blow to intellectual freedom and personal privacy imposed by clauses in the USA PATRIOT act have librarians absolutely steamed! The fact that the basic mechanisms of file formatting, that make such full-text databases possible, are unstable and under attack by the same commercial and political forces, is *not* being missed by this group of people.

So when I began to research the task of applying the free-licensing model, which has worked so well for software, to the design processes needed for colonizing space, as we are doing at Anansi Spaceworks on the Narya Project, I immediately realized that stable, free data formats would be a necessity. Experiences with software like Microsoft's Word, Autodesk's AutoCAD, and RSI's IDL had shown me that vendor lock-in was a sure-fire way to kill any free development prospects.

Free design projects also involve a lot of different types of data to exchange: rich-text documents, yes, but also slide presentations, illustrations, software packages, 2D Computer Aided Design (CAD) drawings, 3D CAD models, Computer Aided Manufacturing (CAM) and Computer Numerical Controlled (CNC) machine control scripts, audio and video recordings, and a miscellany of less common data types.

What I found, is that the results are somewhat mixed. Some content types have good and obvious free-format choices, some have only proprietary formats or very poor free formats that can't compete, and still others are engaged in pitched battles between free and non-free standards. Each of them tells a piece of the story, and shows what we may expect from the years to come.

The writing is on the wall

The awareness of the free format issue is pretty high, and probably nowhere higher, than with word processing documents. The only serious proprietary contender here is the Microsoft Word DOC format. All of the other formats, including the Word Perfect WPD format are pretty much on the way out, and even Microsoft itself has capitulated to the degree of focusing on its more open RTF format, and promoting XML. Although, as has been argued elsewhere, XML is by no means a sure-fire way to a meaningfully free file format.

Which is not to say that DOC is dead. That would clearly be wishful thinking, as I know from conversations with content providers like the National Space Society, which has consistently used MS Word DOC format in a misguided attempt to provide educational materials in a "common" format. It can be quite difficult to persuade authors and distributors of such information; even that the format is a thing worthy of serious thought, let alone try to explain why requiring all of their potential audience to have the latest version of MS Word to read their work, is a very bad idea.

Among people more in the know, such as librarians and serious researchers and publishers of content, the awareness is growing

Nevertheless, among people more in the know, such as librarians and serious researchers and publishers of content, the awareness is growing. You don't have to go through too many frustrating experiences trying to read files that aren't fully forward or backward compatible to get the idea that something must not be right. That's all while being faced with huge stockpiles of data that must be read from tape or CD and converted file-by-file and rewritten to other media. With this audience, the only real trick is to get them to realize that the problem is the closed format, rather than, say, an intrinsic failing of electronic media. In other words — help them to realize that the problem is artificial and solvable.

The most extreme reaction to this is that of the Project Gutenberg archive, which has opted (at least for most of its existence) to use only ASCII-encoded plain text to store their documents. Of course, this makes the documents much less usable, since only through human intervention is it possible to add the expected text formatting, but it has served them very well.

Acceptance of PDF is very deep: you can get all your tax forms this way, and most government sponsored research reports are released in PDF, or occasionally, in HTML, which can also be regarded as a useful rich-text file format, even if we do generally only associate it with the web itself. And although there are some misgivings about the PDF standard, seeing that it's driven entirely by its originator Adobe, in order to promote a proprietary product, the standard is generally considered open since it continues to be documented by a published specification.

In the technical science and engineering communities, of course, the older $\text{T}_{\text{E}}\text{X}$ and $\text{L}_{\text{A}}\text{T}_{\text{E}}\text{X}$ standards (which are definitely free) continue to be prominent. Combined with XML

based markup systems such as Docbook and MathML, and converters to Postscript and PDF formats, we have a fairly complete system for academic authors using these content-aware text-formatting systems.

In the more conventional word processing world, of course, there is another open standard, which has emerged, based on the OpenOffice.org project, and standardized by the OASIS standards body. This standard is not yet so widely accepted, but there is some likelihood that it will be.

With the caveat that we will probably need import methods for DOC and RTF formats for some time to come, it's pretty clear that the battle for richly formatted text data will be won by free formats. There's a lot of awareness towards moving the whole of business and technical communications forward to free and stable file formats. And this is not only in the free software community, but in a much broader community of technically-inclined producers, who are valuable as opinion leaders and early-adopters.

The same old song and dance

Exacerbated partly by the brouhaha over internet file-sharing and the MPAA and RIAA's ham-fisted response to this perceived threat, the formats for interchanging and storing audiovisual data have received a lot of attention. This is something of a red-herring, but it has at least drawn developer attention to the problem.

There are at least two serious cases of patent-encumbrance to be found here — in the cases of the GIF image format and the MP3 sound-recording format. In both cases, software patents were used to insist that programs implementing these standards were automatically subject to royalties or other limitations. Although lax enforcement prevented either from being a really serious threat, these incidents exposed the potential threat of such patent-based attacks.

The free software community rallied very well in both cases - inventing the Portable Network Graphics (PNG) and the Ogg Vorbis (OGG) standards respectively. Both standards are technically equal or superior to their proprietary competition, and stand a good chance of taking over in the transmission of such data on the web. It's very likely that the lax enforcement of patent protection for both GIF and MP3 was motivated in part because of the ease of migration to PNG and OGG formats.

Dark and disturbing things are happening with the storage of music on fixed media, however, because of all the players I can find in my local Walmart, none supported the free Ogg Vorbis format, providing only MP3 and a particularly

nasty new proprietary format from Microsoft called Windows Media Audio (WMA), distinguished primarily by its strong support for "digital rights management". This of course, exposes a new vendor lock-in tactic, which is software/hardware cronyism: the producers and sellers of digital equipment are highly vulnerable to the hard-ball business tactics that companies like Microsoft are so well-known for. The only really widespread standard for digital video is the DVD standard, which is built on MPEG2 encoding, which is at least a well-documented format. However, Ogg Theora has been started as a true free format for compressed video, to stand alongside Vorbis.

The dominant format for presentation of graphics on the web remains conventional HTML used with animated GIF images, although a proprietary format, Macromedia Flash, has really threatened to take over. Nevertheless, there is gradual progress on the Scalable Vector Graphics (SVG) standard, which, combined with Javascript, and given real browser support, promises to do many of the same tasks — an XML format backed by the Worldwide Web Consortium (W3C).

Planning for the future

After all of this good news, however, entering the world of computer-aided design and manufacturing is something of a disappointment. CAD software has always been very expensive and targeted to relatively few users, tightly bound to the manufacturing industries it supports, and the licensing cost has gone without much complaint, seeing as it sits alongside many more expensive hardware capital costs.

At the very highest levels of industry and government, there has been a definite recognition for the need for CAD interchange formats, and there have been standards such as IGES and the much newer STEP (ISO 10303-1) for doing that. IGES was somewhat limited in the types of data it could exchange, and although STEP is in principle much more capable, it is so complicated that no *full* implementation, proprietary or free, yet exists. You might say that STEP is actually a set of dozens of CAD drawing standards in one unified framework. Moreover, STEP is actually only a schema, leaving the actual data representation open. An XML-based storage format is reportedly under development, but is apparently not complete.

Nevertheless, these standardization efforts are promising, and there is at least one attempt at a free software implementation capable of reading and writing STEP data, called Open Cascade.

This situation is pretty daunting, both for the small-time CAD user and for the free software CAD developer. So it's not surprising that for many, the only seriously used format remains AutoCAD's DWG, which is proprietary and undocumented. Autodesk once promoted the DXF format, currently supported in free software by QCad, for drawing exchange, but it is considered too limited a format for serious CAD work. Also worthy of note is the XML format for 2D CAD used natively by PythonCAD.

It might be worth considering whether free software developed for computer graphics modeling, such as Blender, might be adaptable to 3D CAD applications, although it's clear that this might be far from trivial. Nevertheless, Blender provides a native format, which may be regarded as free, seeing that a free implementation exists.

Interested developers should probably check out one or more of these projects to see what can be done to create a useful 2D or 3D CAD standard, suitable for free software users and would-be free hardware designers to use.

A taxing situation

Perhaps the saddest example of proprietary lock-in gone mad, government cluelessness, and blind support for vendors to the detriment of the people is the IRS "e-file" system.

The goal is a very laudable one — simplify and reduce the IRS's costs in processing tax returns by making the entire process electronic. But of course, somebody gets hurt by this: the large number of commercial tax preparation businesses, whose business is driven entirely by the difficulty of preparing taxes to comply with the US's constantly changing income tax laws. As a result, after the government's usual process of working with "stakeholders", we have a completely proprietary e-file system, which locks users into using commercially provided, proprietary tax preparation software, if they want to get the advantages of e-file.

My latest tax return booklet tells me I'm probably eligible to use "free e-file software". However, not only is the software non-free, but it relies on services which demand that you agree to having your personal income tax information used for marketing purposes. It's hard to imagine a more complete disregard for personal privacy! As a result, I still file my taxes on paper, and I'll do so until they get it right.

What's "right"? Probably adopting a standard like the Tax ML standard in design at OASIS, precisely for this purpose. It's hard to imagine any natural reason for the IRS wanting anything else: an XML standard for expressing your tax re-

turn would allow for easy automatic verification and cross-checking, and the whole process could be easily and completely automated. This exposes another proprietary lock-in weapon, though: the government is often completely clueless about the importance of free interchange standards for promoting a free market, instead swallowing industry propaganda, which promotes proprietary standards.

The government is often completely clueless about the importance of free interchange standards for promoting a free market, instead swallowing industry propaganda, which promotes proprietary standards

Unlocking the door

From all of these examples, we can see the methods, which have been used to create and promote proprietary formats and the vendor lock-in:

- The oldest trick is the one that initially made the DOC format so difficult to follow: simple obfuscation. Don't document it and change it constantly so that it's hard to reverse-engineer.
- The next twist is to claim a software patent on the format. You can do this because there are essentially no standards for denying software patents, leaving this a complete loose cannon in all sorts of situations — including file formats.
- Next, the use of encryption technologies, ostensibly introduced to protect the user's file data, but implemented in ways that also obscure the format. This wouldn't be much of a problem, except that the DMCA has made it a criminal act to decrypt such a file and reverse-engineer it, under some interpretations of the "circumvention device" language it includes.
- Business-to-business dealings can deal free formats out and closed ones in — just like the consumer audio equipment that can read the hardly-used proprietary WMA format and not the free OGG format.
- Sadly, the government practice of consulting "stakeholders" when making policy, tries to "promote commerce" by essentially trying to do whatever the industry tells them to do, without serious consideration of the *public* stake in the technology.

The first three problems are problems with using proprietary formats — to be avoided by using free formats instead.

In order for this to work, though, the free software community has to ensure that free standards do exist, either through standards bodies, or through well-documented *de facto* commercial standards. We also need to settle on a widely accepted definition of what a free format is — I can suggest our own Texas OSI's definition and the guidelines put forth by the Open Data Format Initiative as starting points for that, but there is no widely accepted "Free Format Definition" to compare with the FSF's Free Software Definition, which is a major stumbling block in promoting free formats.

The last two are bigger problems — the means by which free formats can be shut out: either by corporate cronyism, or by government agency fiat. Greater support for free hardware development may address the absence of good hardware for handling free formatted data, but the government problems can only really be fought with legislation.

Fortunately, free format laws, should be, as has been pointed out by the Free Software Foundation and the Electronic Frontier Foundation, much easier to defend than laws to preferentially support free software. The nature of public documentation makes it clear that transparency, public design accountability, and openness to public audit should be goals for the government of any free nation, and this is the tack to take in promoting free formats for government documents. So far, the proprietary world has yet to come up with a serious answer for that — the value of free formats is simply too obvious to defend against.

We are not alone

In the end, the free software community is still too small to directly make the kind of widespread change that serious adoption of a free format requires. So, in addition to creating viable standards, we also have to promote the use of the formats to the groups that need them most, and are most willing to adopt them: technical, non-developer users. Scientists, mathematicians, and engineers have a long-standing relationship with free-software, so promoting free-formats to them will be preaching to the converted.

More recently, the troubles that librarians face are forcing them to examine the issues related to file-formats more carefully. They don't always realize that this is the right way to frame the problem, and that's what we need to communicate. It's very important to show that the problems are *not* intrinsic to all electronic formats, only to proprietary ones.

Most of all, there needs to be an openness in the members of the free software community, to speak to people from other disciplines about these key issues which are of benefit to us all. That hasn't always been one of the strengths of the community, but it's probably what's required.

The problems are *not* intrinsic to all
electronic formats, only to proprietary
ones

Once such power-users of file formats are converted to using free formats, they will have a very nonlinear effect on promoting change. After all, the biggest reason people adopt formats is: that the format is what they need to access their favorite information. It's only after a format has come into wide use that it begins to encourage other authors to switch to it. So promoting the idea to technically-capable early adopters just makes sense. But we have to be aware that these users need more than just a new word processing format — technical users have a wide range of data format needs.

Finally, we have to have patience. It won't happen overnight, and most disciplines move more slowly than the computer industry does. But it seems very likely that it will happen. And it must — after all, the inability to share information, driven by superstition and guild secrecy (the first incarnation of so-called intellectual property), is what put the "dark" in "Dark Ages". None of us want to go there, and that's the point we've got to sell as a community.

Bibliography

- [1] MARC 21 Standard (<http://www.loc.gov/marc/>)
- [2] FRBR Standard (<http://www.ifla.org/VII/s13/wgfrbr/wgfrbr.htm>)
- [3] Dublin Core Initiative (<http://dublincore.org/>)
- [4] DMCA information (<http://anti-dmca.org/>)
- [5] Public Knowledge (<http://www.publicknowledge.org/>)
- [6] Anansi Spaceworks (<http://www.anansispaceworks.com>)
- [7] Narya Project (<http://www.narya.net>)

- [8] Project Gutenberg (<http://www.gutenberg.org/>)
- [9] OASIS Standards Body (<http://www.oasis-open.org>)
- [10] PNG Standard (<http://www.libpng.org/>)
- [11] Ogg Vorbis (<http://www.vorbis.com/>)
- [12] Ogg Theora (<http://www.theora.org/>)
- [13] Scalable Vector Graphics (<http://www.w3.org/Graphics/SVG/>)
- [14] IGES CAD format (<http://www.nist.gov/iges/>)
- [15] STEP CAD format (<http://www.steptools.com/>)
- [16] Open Cascade library (<http://www.opencascade.org/>)
- [17] PythonCAD project (<http://www.pythoncad.org/>)
- [18] Blender program (<http://www.blender.org/>)
- [19] Texas OSI free format definition (<http://open.narya.net/>)
- [20] Open Data Format Initiative (<http://odfi.org/>)

Copyright information

© 2005 by Terry Hancock

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.2 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license is available at <http://www.gnu.org/copyleft/fdl.html>

About the author

Terry Hancock is co-owner and technical officer of [Anansi Spaceworks](http://www.anansispaceworks.com) (<http://www.anansispaceworks.com>), dedicated to the application of free software methods to the development of space.



A PLANT NEEDS WATER TO GROW

THE OPEN VOICE NEEDS SUBSCRIBERS TO PROSPER!

Free Software MAGAZINE

BY SUBSCRIBING YOU WILL BE SUPPORTING A MAGAZINE WHICH BELIEVES IN FREE SOFTWARE. ALL OUR ARTICLES ARE RELEASED UNDER THE GNU FREE DOCUMENTATION LICENSE, ENHANCING EXISTING INFORMATION ON FREE SOFTWARE.

SUBSCRIBE NOW!

[HTTP://WWW.FREESOFTWAREMAGAZINE.COM/SUBSCRIBE](http://www.freesoftwaremagazine.com/subscribe)



Center of the System Administration
www.sbin.org

Open look at common things



Go Open

Open Source Software, which originally developed out of the passion of hackers and geeks, has now become the basis of real business solutions. Low TCO, high reliability, security and performance of Open Source software make the shift from "closed" technologies and standards to modern, open solutions inevitable.

CSA offers development, deployment and support services based on Open Source technologies and standards.

Up-to-date solutions for small and medium size businesses are available for you today with guaranteed full 24/7 support and no license fees.

Today, being closed means being left behind

+7 8793 365584

Creating Free Software Magazine

A long path that takes us to the very beginning of this project

Tony Mobily

This magazine was inspired by a conversation I had with a great friend of mine called Massimo. I said to Massimo “I think it would be great to start a magazine. It’s my ideal job, and I think I know what the world needs right now. It’s a pity there’s no money in publishing, and I’m not willing to run a magazine that doesn’t pay its contributors *well*...”. His answer was very simple: “Tony, there’s money everywhere, as long as you do something good and promote it well”. Well, seeing that he has a successful business, I thought I would listen. And it’s thanks to him that you are reading this article right now.

A few months ago, just before my conversation with Massimo, I realised that the world needed a magazine on free software. My ideal magazine would be aimed at IT professionals (the *techs*) as well as managers.

Massimo said “Tony, there’s money everywhere, as long as you do something good and promote it well”
Well, seeing that he has a successful business, I thought I would listen

The magazine would contain technical articles, but they would be focused on describing the possibilities of free software, rather than the technical details of how to configure a specific server. It would also publish technical articles on software patents, copyright laws, and how the world is changing thanks to free software. Above all, it wouldn’t be yet another technical Linux magazine (there are plenty of them at the moment) and it wouldn’t compete with Linux

magazines. It would also break the common rules for magazines of this kind, and contain a fiction section - short stories about the new technological era.

This magazine would pay its authors well, and would release all the articles under a free license.

A few months ago, after talking to Massimo, I decided that I would do it - and I have.

Starting up

When starting a magazine from scratch, the first concern is creating an appropriate structure to support the project (an office, the staff for the magazine’s composition, the managing editor, the web designer, etc).

I have been working on magazines my whole life. I am aware of all the processes involved, and I know that if I had followed the “normal” path, I would have needed a lot of capital to start the project, and a lot of advertisers and subscribers to keep it going. Seeing as Free Software Magazine would attract a restricted audience (we are not *Cosmopolitan* or *Playboy*...), such a structure would have been far too expensive.

There is also the technological side of the story. I wasn’t willing to accept that Free Software Magazine (I will call it “FSM” from now on) would need a compositor to actually make up the magazine *by hand* for every issue. The manual composition of a magazine and the subsequent quality checks take phenomenal amounts of time and money (for a while I was the man who checked that all the captions were correct, all the borders aligned, and so on for another magazine). Also, I wasn’t willing to accept that in order to create FSM I would need proprietary software (I discovered

later that I'd have to give in on this one, but only marginally and temporarily).

A magazine that talks about free software and solutions had to be set up in such a way that composition simply wouldn't be needed: the magazine's PDF and HTML versions would have to be generated *automatically*, providing the articles as input.

Well, I can now say (and not without immense satisfaction, and a sigh of relief): we did it.

The system we created can now take an article written using OpenOffice or Microsoft Word (using the right styles), and generate the XML version - as well as an extremely professional looking PDF file.

It has been an up-hill struggle. I have designed the whole system, coordinated the amazing people who wrote various components, and coded a great deal of it myself. It has been hard. I have sent and received more than two thousand emails for this project. But in the end, we did it. The system is here, and we are now using it for the real "Issue 0" that I am writing right now.

The initial technical planning: XML

A magazine is a collection of articles. Deciding the format for the articles was, in my opinion, one of the most crucial steps of the project - everything else would depend on it. Getting it wrong could have compromised the project's success.

The choice went automatically to XML: it's a language that allows you to define your own file format; it has existing, powerful tools (such as XSLT and XPATH) for converting data into HTML and other formats; it is supported by every platform on this planet; and (which is quite important as well) I had worked on it before, even though it had been a while.

But it's not enough to say, "I'll use XML files", there's still all the design work that needs to be done.

XML lets you decide what tags (or, more correctly, entities) you will use in your mark-up file. You normally do this by writing your DTDs (or more modern schemas). But this decision is an important one, as it's very easy to design an XML structure that simply doesn't work properly. What's worse, you may discover the problem further down the track, when changing a detail in the XML file generates a chain reaction that explodes into many, many hours of work.

This is why the first thing I had to decide was: do I want to use an existing, established DTD, or shall I define my own?

I did my research, which was crippled by my limited knowledge of XML; I looked into other systems that dealt with similar issues, but they all looked too complex or boring to me.

I wanted a simple, lightweight XML structure that would contain exactly what I needed - after all, if XML was a way to store information intelligently, who could decide what information to store better than me? I had worked in the industry long enough to know what information I needed for each article. So, I designed it.

Well, I did it with the help of Michael Eastwood, who has three fantastic qualities: he's a genius, he knows XML very well, and he's a graphic designer. Michael did a lot more than help me with XML: he designed the initial web site, and wrote the XSLT transformations to translate articles into HTML (Michael said from the beginning that it would be up to me to set the XML structure).

Here is what a very basic article looked like:

```
<?xml version="1.0" encoding="UTF-8"?>
<page>

  <title>
    <main>The title</main>
    <sub> The subtitle</sub>
  </title>

  <article_info>
    <publication_date
      day="31"
      month="12"
      year="2004" />

    <authors>
      <author>First Author</author>
      <author>Second Author</author>
    </authors>

    <article_type>Article type</article_type>
    <biography>First Author is this.
      Second Author is that.
    </biography>
  </article_info>

  <contents>

    <p>This is a sample article</p>
    <heading>Today's heading:</heading>
    <p>This is a simple paragraph
      under an heading</p>

  </contents>
</page>
```

The structure was very simple: everything is enclosed into a `<page>` tag. An article had a title (divided into main and sub), a publication date, an author, a type, an author's biography, and the text (which was made up of paragraphs).

This is what the resulting HTML looked like after the transformation (please remember that this is a very stripped down version of it):

```
<html>
<head>

  <title>FSM - The title</title>
</head>

<body>

<p class="article_type">Article type</p>
<br>
<p class="date"> Date: 31/12/2004</p>

<h1>The title</h1>
<h2> The subtitle</h2>

<p class="author">By First Author, Second Author</p>

<p>This is a sample article</p>
<h3>Today's heading:</h3>
<p>This is a simple paragraph under an heading</p>
<h3> About the author </h3>

<p class="biography">First Author is this.
                        Second Author is that.</p>
</body>
</html>
```

I created a directory structure that would contain everything; I placed the HTML documents into `htdocs`, and decided that each directory would only contain one XML file called `index.xml` I also created the directory `bin`, where I placed the command `ov_make_html`. This command basically runs:

```
xmllint --xinclude $OV_PATH/xslt/page_html.xsl \
| xsltproc - index.xml > index.html
```

The directory tree you can see in the downloads section of this article is a little more complicated (and so are the scripts), but it's easy to find your way once you understand how the whole system works.

Once all this work was done, I candidly asked Michael: "Why don't we use this fantastic system to create the *whole* web site?" Michael was sceptical. He said "you can if you want, but then you are limited: you won't be able to put anything fancy on the web site, only what's allowed in articles".

There is one thing I hate about Michael: he is always right. However, in that particular instance I was right too: using the same template for articles and web pages would simplify the web site's management to a great degree.

What was the difference between a web page and an article? Well, all (and *only*) the information within

`<article_info>` was inappropriate for a web page (author, publication date, article type, and biography). Therefore, a web page would share the same structure as an article, but without the entity `<article_info>`.

This is why both an article and a web page start with `<page>` (rather than something like `<article>` or `<web_page>`).

This was only the beginning. The hard part had yet to come.

Overcoming the limitations: exec filters

In the previous section, I mentioned that using the same converter for articles and for our web site made the latter look really boring.

One of the main technical decisions I made was that even if the site's HTML was going to be generated automatically, the web site itself had to be static. There are several reasons for such a (possibly limiting) choice. A static web site has the following characteristics:

- it can be hosted anywhere;
- it takes less processing power to serve a static pages;
- it is much, much less prone to DOS attacks, SQL injections, and so on;
- it is easier to maintain;
- it will never return an error message to the clients;
- it is much easier to have it mirrored without going insane with `mod_rewrite`.

So, FSM was going to be an automatically generated, static site.

This created a number of flexibility problems. For example, I wanted every web page to show its "path", so that people visiting it would never get lost while browsing. I truly dislike web sites that have an unclear and illogical structure (and yes, this does mean that I dislike most web sites on this planet...).

But I wasn't happy to put the path information on the XML file itself: what if the file changed its position in the file system? If the PATH information were in the XML file, I would have to manually change the page's XML as well - unfortunately, I dislike duplication of information as much as illogically structured web sites...

I needed a way of embedding some pseudo-dynamic material in a web page. I also needed to prove Michael wrong, and show him that yes; it was possible to make the web site look great. The basic idea was that the result of a program would be embedded into the resulting HTML page; this would give me a great deal of flexibility.

I didn't find any information on how to do this using "standard" XML. Besides, I needed to be able to:

- run a program which would return XML code, which would then in turn be processed by the XSLT processor (a "pre-processor");
- run a program which would return HTML code (a "post-processor").

This is why I wrote the scripts `ov_pre_exec_filter` and `ov_post_exec_filter` (which are really both the same script, but it changes its behaviour according to its name), which I placed in the `bin` directory of the project.

The script `ov_pre_exec_filter` is very simple: it looks for the XML entity `<exec-pre exec="PROGRAM"/>`; it then executes "PROGRAM" and substitutes the program's output with the entity itself. The `ov_post_exec_filter` does exactly the same thing, but it looks for the entity `<exec-post>` instead.

In practice, this would mean for example, that fixing the PATH problem (see above) simply required inserting this entity in the XML file: `<exec-post exec='ov_exec_path'/>`.

The script `ov_exec_path` would take, as input, the path of the page it refers to, and would print out a complete clickable version of the path.

The exec filters system became much more crucial than I expected. It gave me an incredible amount of freedom, and allowed me to make the web site far less boring.

One fine example is the organisation of the "pills". I wanted to keep the web site "alive": FSM is a magazine, and not a news site; as a result, the only news is when a new issue comes out, and updating the web site only once a month can make it a little "static". For this reason, I wanted to have a system where I could publish "pills": short articles about absolutely *anything* (even fiction) that could possibly interest the magazine's audience.

This is what the "pills" page looks like:

```
<contents>
  <p> Every day (well, nearly every day)
  FSM publishes a <i>pill</i>,
  a short article on Open Source's
  "current affairs". </p>

  <heading>Index by the year</heading>
  <exec-pre exec="ov_exec_pill_year"/>
</contents>
```

That's all! The script `ov_exec_pill_year` will scan the directory it's in, and give a list of years to choose from. The

same applies to `ov_exec_pill_month`, which gives you a list of months, and `ov_exec_pill_day`, which lets you choose what pill you'd like to read.

The exec filters also solved the great problem of the copyright notice, which changes depending on the position of the article (articles in `current_issues` are not yet released under a free license).

So, in the XSLT transformation file `page.html.xsl`, you can see `<exec-post exec='ov_exec_copyright'/>`, which is expanded into the right copyright notice for that particular article. The script is very smart: if it finds the "LICENSE" file in the directory where the article is placed, then the copyright notice is taken from the "LICENSE" file. This means that I can decide the default copyright (the GFDL if it's in `free_issues`, copy reserved if it's in `current_issues`), or assign specific copyright for a specific article. This is specifically important for articles in the "fiction" section of the magazine, which are normally not released under the GFDL.

Converting OpenOffice and Word document into XML

After managing a magazine for a few weeks, you discover that it's pretty hard to get an author to follow the writing guidelines.

After a few years, you realise that you were wrong. It's not hard... it's *impossible*.

You also realise that very few people want to write their articles using XML tags. This is the reason for authors writing their articles using OpenOffice, StarOffice or Word.

The point was: how could we let people write their articles, save them in RTF, and then convert the RTF file into our XML?

There were several answers to this question. The main ones were to:

1. open the RTF file in OpenOffice (in batch mode?), save as OpenOffice's native format (which is XML). Then, apply a XSLT transformation to convert OpenOffice's XML into our own;
2. open the RTF file in OpenOffice or Word, and then use a Basic macro to create the XML file by hand.

The first solution would have been better from a technological point of view, and it would have been a lot faster. The big problem was time. Gian Maria, a Microsoft VBA guru,

was available and offered to write the converter - but it had to be a Microsoft Word script.

Writing it wasn't easy at all: we had to decide on a style that would let the authors create a simple, readable RTF document, and convert it into a valid XML file. The solution was (obviously) the use of styles, but we couldn't be too invasive towards the authors, there had to be as few styles as possible, and the RTF document had to be easy to write.

We decided on an optimal file format (which of course was changed a million times in the process) and wrote a finite state automaton that scanned the RTF based on those styles into a valid XML documents.

It would take a long list of articles to explain all the problems we had to solve. In this article, I will only list a few of them:

- API incompatibility between Word for OS X and Windows. I would constantly get back to Gian Maria, saying "you can't really use this function because it's not supported". I don't know how he put up with me...
- UNICODE support. We had to allow UNICODE characters, and the only way to do that was to decode them ourselves (and therefore create the right XML entity). The problem? Think of PCs and Macs... big endian... little endian...
- Speed issues. VBA is very, very slow. The first versions of the converter would take up to 3 minutes to convert a document.
- Text boxes within the articles. The problem with text boxes was that you couldn't use a style for them, because they could contain anything else (headings, bullet lists, etc.).
- Word kept on crashing (especially on Mac).

Gian Maria and I exchanged a ludicrous number of emails. The first document generated by the converter looked OK, and yet it returned about 10 pages worth of problems when checked against the page's DTD. To make things more fun, I would sometimes change the XML format under Gian Maria's nose; some other times, I would report consistency problems in the XML files generated by his macro.

In the end, the converter was rock solid, reasonably fast, reliable on the XML, and above all it *worked*.

I don't think I'll ever be able to thank Gian Maria enough for what he has done. He is a busy consultant, and gave up a lot of his free time to write this piece of software. When he offered to write an OpenOffice version of it I simply ran out of ways to say "thank you"! This time, there is no hurry and the XML format is not going to change. That will be

the next step and by that point FSM will be entirely based on free software.

The PDF generation

There isn't much I can say about the PDF converter which is responsible for converting a bunch of XML files into a fantastic magazine. Gianluca Pignalberi did it all, and I don't know $\LaTeX$ in the slightest!

Without Gianluca, nobody would have ever been able to see a paper version of FSM. In the future, I am sure Gianluca will honour us with a paper on how he created the $\LaTeX$ class for FSM

What I do know, is that he wrote a class called "openpaper.cls", and that his PDF files were simply fantastic. The PDF files created by Gianluca's class don't look like the "typical" $\LaTeX$ journals you see around. The PDFs look like a "proper" magazine, with coloured text, textboxes, and everything else. His $\LaTeX$ class creates a fantastic table of contents and a great looking editorial board box.

His work has been invaluable. Without Gianluca, nobody would have ever been able to see a paper version of FSM. In the future, I am sure Gianluca will honour us with a paper on how he created the $\LaTeX$ class for FSM.

One thing I can say: I wrote the converter from XML into $\LaTeX$, and realised how hard it is to do it! The problem is that $\LaTeX$ wants you to escape particular characters. For example, \$ has to turn into `\$`. It would be possible to do so using regular expressions. But you can't! The first reason is that characters like `\` have to turn into `\$backslash$` (which contains itself a \$ symbol...). The second reason is that these rules don't apply in a CDATA section - and I had used CDATA sections for example in listings... This time, it was me who had to write a finite state automaton, in order to deal with all the escaping required by $\LaTeX$. The script is called `ov_latex_preprocess`, and it applies to XML files before they are turned into $\LaTeX$ files.

The current problems

So, you may ask: what's missing?

Well, first we have to thoroughly audit our code. In general, the scripts tend to handle problems pretty well, but I am pretty sure I put one or two `die()` statements here and

there. I would like to be sure that each script handles critical problems the same way.

Secondly, the system needs to get rid of the need for Microsoft Office to convert RTF documents into XML. This is possible, and it is an absolute priority. Now that we have a working system, and we are in no hurry anymore, we can take the time to “do it right”.

We could for example port the macro to OpenOffice and see if we manage to get OpenOffice to process the file in batch mode; or we could write another XSLT transformation to convert OpenOffice’s XML into our own.

Thirdly, the system needs a graphical interface. Right now you can only use it by the command line, but the system’s inner structure is so consistent that writing a graphical interface would be extremely simple. The possibilities are amazing: you would be able to give your authors a login and a password to access their “slice” of their issue; they could then upload the `index.rtf` file, and see what it will look like once it’s online or even printed.

To conclude...

Even though the system has a long way to go before it becomes a fully automated publishing system (if it ever will), FSM *works*. There are still a few bugs to iron out, but in general everything functions.

FSM’s system is a little bit like Unix: it’s complex as a whole, but each building block is simple and straightforward.

Creating Free Software Magazine gave us an opportunity to work together, and get to know and respect each other more. It is with enthusiasm that we now work on this project, and it will thrive mainly thanks to the hard work we have put in up till now.

Copyright information

© 2005 by Tony Mobily

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.2 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license is available at <http://www.gnu.org/copyleft/fdl.html>.

About the author

Tony Mobily is the Editor in Chief of Free Software Magazine



Mac OS X: Welcome to the jungle

A look inside the Mac OS X software ecology

Chris J. Karr

If software platforms are habitats, the Mac OS X platform is surely the jungle. Mac OS X is a modern Unix-based operating system that combines the classic Unix/X11 environment, a modern Java toolset and runtime, the classic Mac OS Carbon framework, and the NextStep-derivative Cocoa framework in an elegant and user-friendly operating environment. This diversity of strongly supported programming options, combined with Apple's modern hardware and operating system, presents developers and users with a compelling platform for producing and using software packages.

The continued success of the Macintosh platform is due in no small part to the different ways that developers from other environments can apply knowledge and experience from other platforms to produce Mac OS X applications. These different development platforms can be separated into a few large groups – Unix-based, Java-based, and the derivatives of classic Macintosh and NextStep platforms. Since developers targeting each of these groups come from different backgrounds and development philosophies, developers of each platform tend to produce significantly different types of applications. For example, developers targeting the BSD Unix portions of Mac OS X are more likely to develop and produce programs found in other Unix environments, such as command-line text tools and interpreters. Java-based developers bring cross-platform applications such as the Apache Tomcat server and IBM's Eclipse to OS X users. Developers specializing in Carbon are responsible for modern incarnations of applications from Mac OS 9 and before, such as Microsoft Office and the Adobe multimedia applications, while developers targeting the Cocoa framework have applied object-oriented principles to create unique types of applications found only on the Mac OS X platform.

To understand the diversity of Mac OS X's programming options, it helps to be aware of the operating system's history. Prior to the acquisition of Next, engineers at Apple were busy working on the next-generation successor to Mac OS 9 codenamed Copland. When this effort, along with others (such as the Pink partnership with IBM and Motorola) failed, Apple looked outside the company to acquire a successor to Mac OS. Be, with its modern multimedia-oriented BeOS, was a favored choice, but Apple ultimately chose Next, with its more mature NextStep technologies as the next Apple operating system. The NextStep operating system had a number of traits in its favor. It was a modern and mature cross-platform operating system with solid underpinnings and a strong developer community.

It had modern and robust networking capabilities. (The World Wide Web was originally designed and implemented on a Next machine.) In a time when Apple was floundering in the computer market and approaching irrelevance, the Next acquisition also returned the visionary (and not uncontroversial) Steve Jobs back to the helm of the company he co-founded years before.

While Jobs' vision and drive are often credited with Apple's resurgence, the Unix and NextStep technology, combined with Apple's new focus on Java and open standards, created an environment where developers combined skills acquired when working on other platforms with Mac OS X's native feature set to create new applications and libraries. In order to bridge the legacy developers' transitions from the classic Mac OS platform, Apple provided the C-based Carbon framework to ease the porting process and minimize transition costs. The effort to accommodate and provide familiar environments for programmers from the classic Mac environment and elsewhere is one of the key factors in Mac OS

Fig. 1: Safari (Apple) is a Cocoa browser using the brushed-metal theme. It uses standard Cocoa widgets and styles, but does not make use of customizable toolbars or other advanced Cocoa UI elements



X's success as a development platform.

One of the interesting results of this integration of various development environments is that different types of software developers brought their different development processes to the Mac. This diversity of processes is directly responsible for the different types of modern Mac software. Larger developers who have produced software since the classic Mac OS era tend to use Carbon-based technologies. Smaller developers writing new applications exclusively for Mac OS X tend to use Cocoa-based technologies. Migrant developers from the Linux and Unix community continue to program to the Unix interfaces and use the BSD and X11-based technologies, while business and open-source developers of cross-platform tools and applications have adopted Apple's version of Java.

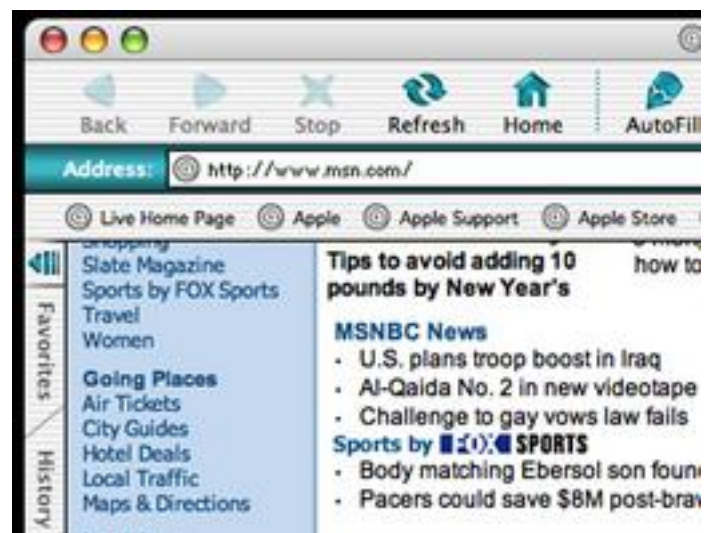
It was a modern and mature cross-platform operating system with solid underpinnings and a strong developer community. It had modern and robust networking capabilities

Commercial developers

The most visible Mac software developers tend to be larger developers. Microsoft, Adobe, Macromedia, and (of course) Apple. They all design and market large software packages for the Macintosh. Microsoft is known in the Mac world for its Office and Internet Explorer products. Adobe

has been active in the Mac community for years with its digital image and multimedia creation tools. Macromedia continues to develop and market its web authoring applications. Apple develops and distributes its iLife applications for casual users in addition to its more professional line of media tools such as Final Cut Pro. Because of the high overhead of marketing and distributing these products via traditional channels and distributors, mostly larger companies occupy the brick-and-mortar shelf spaces. Furthermore, many of these types of applications predate the Mac OS X operating system and consist of significant amounts of code created during the classic Mac OS era.

Fig. 2: Internet Explorer (Microsoft) is a Carbon application. Note the continued use of the heavy pin-stripe theme that was the style of MacOS X prior to 10.3



Since the amount of working legacy code in these products is non-trivial, the producers of larger Mac software packages continue to develop and maintain these products using the Carbon framework. In contrast to the heavily object-oriented Cocoa technologies, the Carbon framework consists of low-level C-based functions and libraries. The use of this framework allows Carbon developers to control basic underlying features, such as Quartz and Quicktime. However, this control comes at a price; the large amount of source code and increased complexity creates an inertia that is hard to overcome when implementing new features or re-targeting the applications to new markets. The primary outcome is that these applications are more complex and full-featured (due to longevity of the product), but these applications evolve slowly and are updated much less often than their Java and Cocoa counterparts.

The continued success of the Macintosh platform is due in no small part to the different ways that developers from other environments can apply knowledge and experience from other platforms to produce Mac OS X applications

While larger developers tend to use Carbon, small independent software developers tend to use Cocoa. Because of the object-oriented nature of the Cocoa framework (previously known as NextStep or OpenStep) and the rapid application development possible with Xcode, smaller developers use Cocoa as a quick route from creating an idea to implementing that idea and making that idea available to interested users. Furthermore, because of the exclusion from traditional channels of distribution due to the overhead involved, smaller developers use the web as the primary means to market and distribute their applications. Since these applications tend to be smaller than their larger commercial counterparts, the market for these applications consists of many users willing to purchase these applications for less money than the larger general applications. Finally, the robust shareware community that the classic Mac platform was renowned for has adapted to this new market configuration.

Because of the smaller codebases, smaller development teams, and lower price points, a rigorous competitive market has emerged where developers compete for paying users. Given that the primary distribution of these products is online - typically in the form of downloadable disk image

files - communication between developers and users is conducted online via e-mail, weblogs, and discussion forums. These factors result in a market where developers are in closer touch with their users. Furthermore, rigorous competition spurs continual development and updates, and new applications are produced daily that attempt to establish new markets. The RSS reader market emerged from such an environment. Although Ranchero's NetNewsWire established the RSS aggregator market, it is currently in constant competition with many similar competitors. This is in stark contrast to markets for products such as Microsoft Office or Adobe Photoshop, which face significantly less competition in their respective markets.

... the Mac OS X platform sports a healthy and growing commercial developer population, it also hosts an equally healthy free software community

An interesting aspect of the small commercial development community is the cooperation between applications in different markets. While applications targeting the same markets compete rather than cooperate, developers will reach out across market boundaries to establish interoperability with other applications. This strategy provides a competitive advantage as sophisticated users can combine different applications to accomplish tasks that a single application would be unable to address. This type of cooperation is evident in the interoperability between RSS aggregators and weblog-authoring tools. In some cases, a developer will offer both types of applications, but still provide compatibility with competitors' products in other markets.

The Java community

Commercial and free software developers creating software with cross-platform compatibility as a feature find Mac OS X to be another viable deployment platform. Commercial software applications have been successfully ported and open-source Java applications have met similar success. In corporate environments where custom Java applications are used to interface with various custom server applications, Mac OS X has proven itself capable of providing an environment that is consistent with the application's goals and requirements, while providing the advantages of Mac OS X.

Console and command-line based applications can often run with few changes. Apple provides a full Java runtime and

software development environment that is accessible via its Bash command line. Graphical Java applications written with Swing run on the Mac using Sun's Steel theme, or Apple's Aqua theme. Apple's Swing-compatible implementation of the Aqua look and feel allows platform-independent applications to run and behave like native applications.

Developers creating software to be run on different Java platforms occasionally discover incompatibilities in areas such as printing or the system clipboard, but these incompatibilities are easily addressed and are not significant enough to deter Java developers from excluding Mac OS X as a target platform.

Free software developers

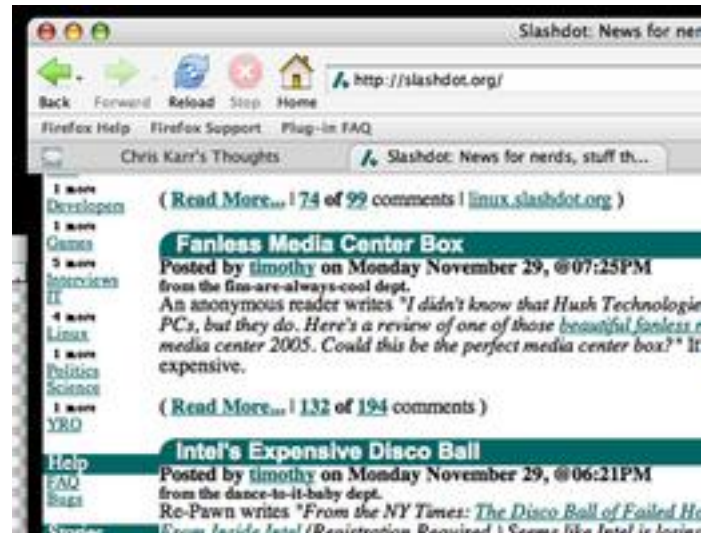
While the Mac OS X platform sports a healthy and growing commercial developer population, it also hosts an equally healthy free software community. In contrast to the Windows platform where free software development is insignificant compared with commercial development, and the Linux/BSD platforms where the reverse is true, free software developers enjoy parity with commercial developers on Mac OS X.

A large catalyst for seeding free software development on Mac OS X was Apple itself. Mac OS X is built upon a free software BSD Unix variant called Darwin. Apple also provides a native BSD command-line environment, which is accessible via the Terminal application. The release of Mac OS X 10.3 included a Quartz-aware version of the X11 windowing system. Apple continues to support free software by integrating the KHTML web engine in the Safari web browser and uses free software applications such as Samba and Apache to provide network file sharing and web hosting. Apple is a very active member in the free software community, and this is expected to continue.

Unlike the Cocoa, Carbon, and Java environments, the command line and X11 environments are populated almost exclusively with free software. A significant portion of this environment consists of Apple's own free software programs and ports, but other free software projects such as the MySQL database and OpenOffice.org office suite target the BSD foundation and primary platform targets often include Mac OS X. Furthermore, projects such as DarwinPorts, Fink, and Gentoo have aggregated and packaged Unix software management tools for Mac OS X. Using these applications, the Mac can install many programs found on the BSD and Linux platforms.

While some developers are content that their applications can run on Mac OS X via the Terminal and X11 environ-

Fig. 3: FireFox (Mozilla) is a port of the Linux and Windows browser. It uses a theme similar to Aqua, but this can be altered using FireFox's theming capabilities. The widgets used within the web pages are the standard Gecko widgets - not the standard Apple widgets

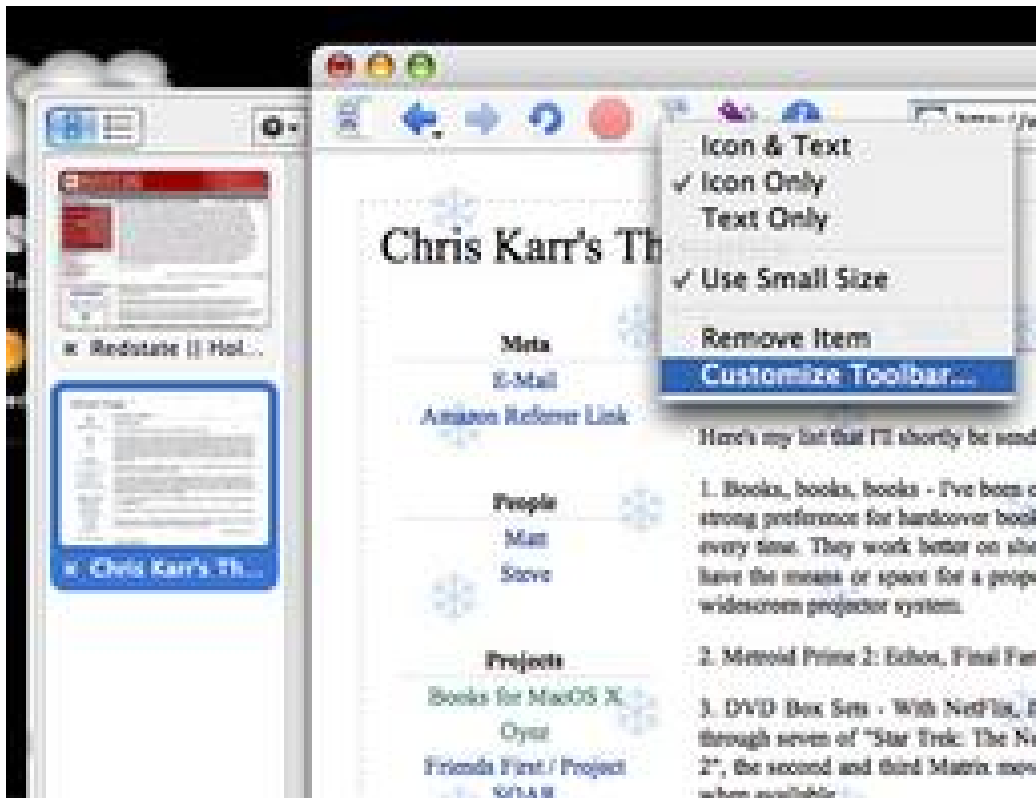


ments, other free software developers adapt open-source GUI applications to take full advantage of Mac OS X. Prominent projects such as Mozilla's Firefox were ported from their X11 and Windows origins to Mac OS X's native interfaces. While some of the Gecko widgets in Mozilla applications are not "Aqua-fied", the icons, application windows, and application packaging follows Mac OS X standards. In addition to the Firefox port, the OpenOffice.org and Gimp projects currently pursue similar porting goals.

In addition to free software originating from other platforms, Mac OS X inspires developers to create native applications unique to the platform. One factor motivating developers are the advanced multimedia and network capabilities of Mac OS X that Apple exposes via the Carbon and Cocoa frameworks. The other factor in the motivating developers is the availability of free developer tools and documentation. Unlike Windows development tools, Xcode and related applications are available online via Apple's website and Apple bundles these with Mac OS X install disks. Furthermore, while O'Reilly publishes books like "Learning Cocoa with Objective-C", Apple makes the same content freely available online. Free tools and documentation enable new developers to start developing full-fledged Mac applications quickly.

The free software native Mac OS X community mirrors the small and independent developers in many ways. The same factors that motivate small developers to flock to the Co-

Fig. 4: OmniWeb (Omni Corp.) is a pure Cocoa browser using the standard Aqua theme. Note the use of advanced Cocoa functionality such as drawers, customizable toolbars, and the WebKit rendering engine



coas also attract free software developers. In some cases, free software applications compete directly with commercial counterparts. For example, the open-source Adium instant messenger client competes with iChat and the Proteus clients. Another example is the Camino project that seeks to produce a fully Cocoa implementation of a Gecko-based browser to the Mac. This product competes with browsers from large developers, other free software projects, and independent developers (Internet Explorer, Firefox, and OmniWeb, respectively).

Other native free software applications carve out their own new niches. The Growl notification engine is one such application. Furthermore, free software applications also cooperate with other free software and commercial applications to provide enhanced functionality for their users.

If the number of new applications and the updates reported on sites like MacUpdate and VersionTracker are any indication, software development on Mac OS X continues to grow and progress. As Apple continues to attract new users with their hardware and software, its development options will continue to attract developers with new ideas and energy to produce new software for the Mac. If recent trends continue, there is no reason to think that the diversity, qual-

ity, and quantity of Mac OS X applications will decline any time soon.

Welcome to the jungle.

Bibliography

- [1] Linzmayer, Owen "Apple Confidential 2.0", No Starch Press:2004

Copyright information

© 2005 by Chris J. Karr

This article is made available under the "Attribution-NoDerivs" Creative Commons License 2.0 available from <http://creativecommons.org/licenses/by-nd/2.0/>.

About the author

Chris is a software developer for Northwestern University and develops the open-source Books library management software in his free time.

The magic of live CDs

What are live CDs, and how do they work?

Harish Pillay

A

“Live CD” is a bootable CD, which contains pre-configured software, this allows the user to be productive without accessing any other hard drives (unless the user wants to store information).

Why would anyone want to have to carry around a CD, rather than having a desktop or laptop computer, which is fully installed and ready to go?

The value brought by live CDs is not immediately obvious to the majority of users, who have only known the reality of going through an installation process (or factory pre-installed), powering up the machine and using it.

Consider the scenario of wanting to purchase the top of the line computer (whether laptop or desktop) from an online store. You have chosen the best combination of hardware for your money and you get it shipped. The machine arrives and you drop in your live CD - in 30 or 45 seconds, you are up and running. You use the hard disk (if any) as your storage medium only, and when you log off, you remove the live CD and put it away knowing that if your machine is ever stolen, the data in the internal hard disk is useless to anyone for the entire drive is encrypted with your private key and secured. You could also save your information on a USB mass storage device.

From a corporate security aspect, your read-only CD is safe as it cannot be tampered or infected (it is read only after all).

A brief history

One of the earliest Live CDs was in the shape and size of a credit card and it was called **The Linux Bootable Business Card** (<http://www.lnx-bbc.org>). This project continues to thrive (it has now reached version 2.1). The

Linux Bootable Business Card is a working system in 50MB (which is usually the capacity of a credit card sized CD).

A remarkable and popular live CD today is Knoppix, created by **Klaus Knopper** (<http://www.knopper.net>). The project was commissioned by **Linux Tag** (<http://www.linuxtag.org>) and has now reached version 3.7. Klaus Knopper's work has spawned a whole ecosystem of Knoppix-like live CDs with specific editions for different target audiences: bioinformatics, education, computer forensics, gaming and not forgetting the quintessential desktop. It's now entirely conceivable that an individual could have a bunch of these live CDs and use the right one according to his or her activities during the day.

One of the earliest Live CDs was in the shape and size of a credit card and it was called

Almost all of today's live CDs run Linux. I only know of one non-Linux live CD, which uses FreeBSD, called **Freesbie** (<http://www.freesbie.org>). Normally all of the software available on live CD sites are released under a free license (GPL or BSD); however, some live CDs contain non-free software, and that could be a problem if you want to be free to copy and distribute its contents.

Today, we have the **Morphix project** (<http://www.morphix.org>), which builds upon the intelligence of Knoppix, and allows anyone to build a custom live CD with minimal effort. A quote from Morphix's site states:

[Morphix is] a whole operating system, to install your programs on and give out. Why send out in-

stallation disks, give them a whole operating system with your files.

If you're an IT director you could create a live CD built on RPM or Debian, put the relevant applications into it, encrypt it and issue those CDs to your organization. This way, you could migrate an entire organisation and not worry about viruses, spyware, etc ever again.

When you're working on a system, you'll always create data and files that you'll want to store. If you're using a live CD, you have several options. You could use USB-based thumbdrives, available today with 1GB or more. You could also simply store the files remotely via the Internet (maybe connecting to the remote server via a secure virtual private network (VPN)). You could also use the clever technique of saving your files to your **GMail folder** (<http://richard.jones.name/google-hacks/gmail-filesystem/gmail-filesystem.html>)!

Live CD world tour

The freedom on which the free software community is based allowed the creation of an ever-expanding selection of live CDs. Here's a quick review of some of them:

 The freedom on which the free software
 community is based allowed the creation
 of an ever-expanding selection of live
 CDs

Adios (<http://dc.qut.edu.au/>)

Adios is built from the Fedora Core 2. Adios boots up into a system, which looks and feels a lot like the Red Hat Enterprise Linux environment, and includes all of the office automation tools (OpenOffice, Mozilla/Firefox, Evolution etc). However, this was not what Adios was set out for: it was meant to be a teaching tool for learning networking principles and distributed computing. It is a highly recommended live CD because it's based on Fedora Core 2.

Dyne:bolic (<http://www.dynebolic.org/>)

It is developed to meet the "needs of media activists, artists and creative people as a practical tool for multimedia production". With this live CD, you're able to both manipulate and broadcast audio and video over the Internet.

Naturally, you will be able to record, edit, encode and stream audio and video using the devices normally installed

Fig. 1: Dyne:bolic's web site



in modern computers. This live CD is remarkable: in 60 seconds you can get an Internet radio and TV station up and running.

Knoppix (<http://www.knoppix.net/>)

As I mentioned earlier, this is the gold standard of live CDs. Knoppix continues to inspire a rash of remastered versions for a wide spectrum of users and environments.

Slax (<http://slax.linux-live.org/>)

This CD is built using the Slackware distribution - one of the earliest Linux distributions. Slax continues to be true to the Slackware tradition and should be lauded for that.

GeeXboX (<http://www.geebox.org/>)

It is another clever incarnation of Knoppix. This time you're given the ability to turn your computer into a media center and the ability to play DivX, DVD, VCD and SuperVCD (as well as audio) without any extra effort. Connect your TV to the output of your computer and with GeeXboX, you will have a full-fledged video player!

Educational live CDs

No one-paragraph description is going to do justice here. Educational settings are probably where the next wave of innovators will emerge. Here's a short list of projects, which definitely deserve attention:

- SkoleLinux (<http://www.skolelinux.no/>)
- The Open Source Education Foundation (<http://www.osef.org/>)
- Organization for Free Software in Education and Teaching (<http://www.osfet.org>)

- The K12 Project (kindergarten + 12 years) (<http://www.k12os.org/>)
- The SchoolForge (<http://www.schoolforge.net/>)

Conclusions

CD-based systems will grow in popularity, and in my opinion this is a trend that has to be watched.

While it is true that CDs are generally slower than hard disks, the innovative way in which live CDs can bring value to an organization should not be underestimated. Some of the innovations riding on this include the [Stateless Linux project at Red Hat](http://people.redhat.com/dmalcolm/stateless/stateless-linux-HOWTO-en/) (<http://people.redhat.com/dmalcolm/stateless/stateless-linux-HOWTO-en/>) and the [FreeNX project](http://www.nomachine.com/) (<http://www.nomachine.com/>).

Bibliography

- [1] A page which contains several useful links (<http://www.linuxnews.ws/li/livedcd.html>)
- [2] A comprehensive list of live CDs (<http://www.frozentech.com/content/livedcd.php>)

Copyright information

© 2005 by Harish Pillay

This article is made available under the “Attribution-Share-alike” Creative Commons License 2.0 available from <http://creativecommons.org/licenses/by-sa/2.0/>.

About the author

Harish Pillay has been in the computer industry since 1982 having built his first computer, a 6502-based machine in 1980. He is currently employed as the Chief Technology Architect of Red Hat Asia Pacific based in Singapore. Harish helped found the Linux Users' Group (Singapore) in 1993 and continues to be an active advocate of FOSS in Singapore and Asia in general. Harish holds a BS in Computer Science and a MS in Electrical and Computer Engineering, both from Oregon State University. He is a senior member of the Singapore Computer Society and a long-standing member of the IEEE. When he is not tinkering with technology, he spends his time with his wife and two sons, and is an active member of the Seletar Hash House Harriers "a drinking club with a running problem".





Freelock Computing

The

Open

Source

for

Business

Solutions

Open Source is about solving problems. We solve business problems for small and growing businesses in the Puget Sound area. We provide technology strategy consulting, implementation, and hosting. We work with proven open source software. Go to freelock.com for details.

"In today's world of crappy software vendors who provide crappy products and next to zero service at premium prices, it's refreshing to work with someone who is honest, thorough, reasonable and willing to do what it takes to meet the customer's needs."

– Eric Leung, IS Manager, Outdoor Research, Inc.

We Wrote The Book!

"'Open Source Solutions for Small Business Problems' is a very highly recommended book for anyone who is looking at the open source market ... This is easily one of the best Linux books of the year; providing a management level view of the Linux world without the technical focus of other books."

– Harold McFarland, Readers Preference Reviews



Find an open source service provider or sign up as one!
<http://opensourceproviders.com>

Contact us at:
info@freelock.com
1-206-579-4836

Freelock LLC
www.freelock.com

Every engineer's checklist for justifying free software

Free software is not just about “no license fees”!

Malcolm D. Spence

In a few years viewing source code within the major components of software infrastructure will probably be a routine way of doing business. In the meantime it seems that the only reason managers want free software is because it is free (as in free of costs). That's not a good reason in itself: in the long run there are compelling reasons that robust, mission critical infrastructure software should be made free software.

For over 5 years, we at OCI have been supporting free software CORBA products. Clients have been using them to build elaborate integrated software systems. During that time many of our clients have extolled the virtues of free software in regard to meeting their needs. The money to pay for software never came out of their pockets so clearly those virtues didn't relate to price.

Free software slows ever-present market pressure to support only a single platform, or a very narrow set of platforms

It turns out the focus was on them being able to do a better job. If you have been wondering what the impetus is, for switching to free software, from a technical standpoint, perhaps the list we (OCI) have compiled over those years can help. The list breaks into fourteen categories but sometimes the benefits spill into other areas.

Configuration Options and Management

The focus here is on platforms used, compilers supported, etc.

Compiler or IDE choices: You have the source code. This means you can choose your compiler. Your ability to write highly portable code improves when you can create a development environment with a compiler that spans many platforms. Many proprietary products lock you into a single compiler, often linked to the hardware platform. Free software products often support multiple compilers that you can choose from. Keeping code portable keeps your options open.

Upgrade on your terms: If for some reason you do not want to upgrade the platform's operating system, in lock-step with everyone else, you don't have to. You can keep using previous versions and add patches or enhancements selectively. But be warned! Free software projects are often cutting edge. Later versions of the product may require a later version of the compiler for instance, as they stress the compiler support of programming techniques.

An obscure platform need: If you have a special platform that you want the software to run on, then do the port yourself, and create an affinity group within the community to help you spread the load on your maintenance activities. Don't let someone else dictate platform policy for your organization. You don't have to follow the crowd.

Are you a vendor who is feeling left out? If you are a platform vendor and you find your platform is no longer supported in a technology area, or you think it's poorly supported, find a free software solution and support it or sponsor it yourself! Give your customer base insight into how to tune the software to maximize performance. No longer will you have to worry about a software vendor having platform architectural biases (suspected or real!), or casting your platform in poor light.

In general free software slows ever-present market pressure

to support only a single platform, or a very narrow set of platforms.

Free software is an inclusionary style for platforms, versus the more prevalent exclusionary style (i.e. support for only popular platforms), traditionally adopted by software vendors. The result is: free software supports choice and proprietary systems can effectively limit choice. A platform vendor, with a well integrated stack of free software can field a competitive solution, even in a niche market.

The result is: free software supports choice and proprietary systems can effectively limit choice

Revision Management and Product Evolution

Worried about vendor viability? You don't need to negotiate putting code in escrow as insurance for critical projects. You have the source code now, and forever. Normally you can't inspect the escrowed code ahead of time. When the time comes, and you have a need to gain access to the source, it won't be the time to find out how much effort is needed to support it.

Worried about new releases? You can monitor the development activities in the beta code base, bug lists, etc. and measure progress for yourself. No more blind acceptance of a vendor's optimistic delivery dates, that are unlikely to be met. You can even help to pull in schedules, with people or funding.

Want to be a beta tester? Everyone can beta test the next release. It's an open process, not restricted to a privileged few. You can verify its stability and features, get a jump on using it, and then plan accordingly.

It's then released for development, or production use, when the free software development team feels it's ready. It's not just released so that quarterly revenue targets can be met for management. There's less chance of you helping a vendor debug their product when it's not really ready for prime time.

Enhancements

Need a feature? You don't have to wait for the vendor to add features you need. If it is urgent, you can do it yourself. Be warned though you should offer the feature back to the community. You should avoid supporting a specialized version of the code, if you can. Leverage the community.

Test a feature. When you add features and submit them back to the community, a lot of people you don't know, and don't have to pay (but who are very smart) will help you improve it.

Community members can't make bad ideas become good ideas, but they can turn good ideas into great ideas

They can't make bad ideas become good ideas, but they can turn good ideas into great ideas.

You can buy influence! If it's really important, you can also participate in the overall development process. You can influence schedules and priorities by contributing. It's the gift that keeps on giving, as others pick up supporting the project where you leave off.

If it's not that important to add new features, you can still enjoy the benefits. Nobody makes you contribute: use it "as is".

Found a bug? You can patch the code yourself if you need it right away, and post it to the newsgroups or core team to get quick feedback.

You don't have to upgrade if you don't want to. You have the source to support yourself at any release level that you choose.

You don't pay for enhancements that you don't necessarily want. You don't pay, period! No more marketing driven gimmicks masquerading as major features.

Debugging

Don't guess where the problems are: with the source available you can use a debugger to navigate, as your own code interacts with that of the product. In this way you can isolate bugs, both in your code and the product code, faster. Developers consider the access to the source code during debugging to be a huge benefit. The improved productivity is dramatic and substantial enough in itself to warrant consideration for free software as major element of the risk reduction strategy for any major project.

Better comprehension: While stepping through the product code using a debugger, greater understanding can be gained into how the product works. Your code can cooperate with the free software product more effectively when you understand its behavior. You might also pick up some good tips and techniques. Often you are looking at world-class code.

Testing

Faster hardening of code: with access to the source code, people using a wide variety of platforms, operating systems, and compiler combinations can compile, link, and run your code additions on their systems to test for portability. This sort of scale, and parallel debug, cannot be easily duplicated within the confines of a single vendor's testing labs. Testing is the one area of software development that lends itself to scaling.

Testing is the one area of software development that lends itself to scaling

In general, developing for, and testing across, multiple platforms results in robust code. Quirks can be exposed and expelled if possible, or at least isolated.

Documentation

People outside of the core development group are more likely to contribute additional user documentation. Engineers often document something for their own needs and sense it might have value to the community and so add it to the code base. Different perspectives can provide novel solutions. It's not unusual for a few tutorial slides to snowball into a fully-fledged self-paced training class.

Code Usability Issues

It's easier to gain a deeper insight into the behavior of the software, by inspecting the source code, than it is by guessing, or trying to use reverse engineering techniques. (Which may even be considered illegal, by some interpretations of the patent and copyright laws.)

Users are now stakeholders: everyone must succeed or no one succeeds

Really interested users, also known as "Power Users" are more likely to exercise the code during beta activities, as opposed to only working with the product after it's been released. This means there are more opportunities (when the code is still in fluid state) for the user's ideas to be incorporated into the code.

Ownership

Users are now stakeholders: everyone must succeed or no one succeeds.

There is no adversarial relationship between vendor and client. That isn't to say there is no clash of ideas. Anyone who has monitored free software project newsgroups can testify to the lively discussions they contain. However the focus on the product, the openness, the peer review, all make for a good, fast, and candid way get ideas and opinions on the table. Participants learn very quickly, how to present ideas in a clear and coherent fashion in a civilized manner, according dignity to others.

Third-party tools

By having access to the source code, even proprietary tool developers are better able to provide additional tools and add-on products that can enhance the code functionality. Without access to the source code developers must frequently reverse-engineer file formats and APIs.

The tool and add-on environment is on more of a level playing field.

Evaluations

Evaluating free software is easy: no time limit pressures, no salesman calling every week.

Evaluating free software is easy: no time limit pressures, no salesman calling every week

There are no "limited use" rules, restricting the number of evaluators or product features: everyone in your organization, who is interested, can get involved.

There's no legal paperwork to process, or "permissions" to seek. Only the evaluation consumes your time. Informal evaluations are easier. (Do it at home if you feel strong enough about the potential of a free software solution.)

A free software evaluation activity helps you sort your needs, priorities, and get some familiarity with the domain of both the problem and potential solution. If you want to speak with a vendor subsequently, about a proprietary product, you can have that dialogue from a position of strength. You are now an informed buyer!

Benchmarking

Benchmarking is an important activity for determining if a product is right for you.

Remember that many proprietary vendors make you sign agreements that prevent you from publishing or sharing benchmarks regarding their products - particularly comparisons that you might make between their product and other vendors' products. It's an understandably protective position: bad benchmarks can get out, sully the reputation of a product and be expensive to recover from. However no such requirements come from free software vendors! Free software communities have no interest in suing their stakeholders and partners. They're the ones who can help the community in improving the code. The community isn't interested in hiding the results or inhibiting evaluations.

Get those benchmarks out there in the open and let the community assist you in building the right kind of benchmark. Share or improve what's already out there. Free software providers will usually package suggested benchmarks. Build on what's available, this will save you time. Add your benchmark code to the mix. Someone else is likely to improve on it for you. The community will also help you interpret them.

You must measure what matters. You cannot improve what you cannot measure. The more measuring goes on, the more improvement will occur.

Porting someone else's benchmark to your platform and comparing results helps us all understand implementation differences across platforms. This is an important insight for the evaluation process. Does a product mask or magnify the variation that occurs between platforms?

When done right, benchmarking can cost a lot of resources. Free software approaches can provide you with more leverage.

When done right, benchmarking can cost a lot of resources. Free software approaches can provide you with more leverage

Open approaches can help avoid errors in the selection process. Many free software projects include performance testing as part of their regression testing. Performance issues are put in the open to be addressed.

Proprietary vendor benchmarks are often very selective. They have to portray their products in their best light. Free software benchmarks play to the market of free ideas.

Security

Free software means what you see is what you get. You can inspect the code, line by line, to ensure that no disgruntled programmer has buried logic bombs, trapdoors, Trojan horses, viruses or any other nasty surprises in the code.

You don't have to worry that being late with that license fee might result in a locked up system. There are no worries, as with proprietary systems, that the code may contain the means to disable the software, and effectively your business. Free software is not UCITA!

You don't have to worry that the weak link in a security strategy might be some proprietary application with poor defensive measures. You can add your security features to free software, if you wish, and ensure a consistent level of protection across all applications in the system.

Licensing

With free software there are no licensing fees, no development fees, and no runtime fees.

You can put free software where you want it, when you want it. Performance and other considerations drive those decisions, not the licensing model constraints (such as node locks).

There is no arguing with management about money for license upgrade funding just to stay current; upgrading is now mostly a technical decision, not just a financial one.

Are you a VAR? The cost model for your products is more predictable: no more sweating about vendor licensing model changes, which might break your pricing strategy.

There's no vendor trying to use a combination of licensing and proprietary extensions to keep you locked into an implementation.

There are no concerns that your vendor might disappear without a trace leaving you with node locked software, a rollout coming up, and no way to implement more licenses (in other words, you have no way to deploy your application). This may seem remote, but it does happen and has catastrophic results.

Switching costs are not inhibited, or influenced, because of their magnification due to the sheer volume of systems involved.

You can recommend the software to others on its own merits, without worrying about the cost implications.

You can build it into, or ship it with your products, as a way to help your customers, and to improve your product's utility, without affecting your pricing edge (cost of goods sold).

Your own product's functionality can continue to be improved and take advantage of the free software product's improvements without worrying about the repercussions of your customers having to pay upgrade fees. They can stay current with minimal financial impact, or you can ship the new version with your product.

Free software levels the playing field for those smaller companies who cannot negotiate site licenses and other large discount programs that often give larger companies an additional pricing edge.

The special impact of free software on middleware

At OCI, we elected to support free software middleware. We did this for many reasons: we think free software and middleware is a particularly good fit. It facilitates open systems and promotes choice. Object middleware is a special kind of software. Its utility stems from its ability to ensure consistency and interoperability amongst applications with diverse backgrounds and capabilities. The value that middleware offers includes:

- The ability to provide a standard, stable, consistent interface to a wide variety of applications, on a broad set of platforms and enable their inter-operability.
- It decouples service providers from service requesters.
- It enables parallel development of service and client.
- It hides implementation details behind standard interfaces. This allows the implementation to change without disrupting or breaking existing clients.
- It allows different types of clients to share the same services, often simultaneously.
- It frees the developers of distributed systems, from the burden of developing networking software, allowing them to focus on their own application's particular needs.
- It enables the migration of services to new platforms, increasingly diverse and specialized communications technologies, operating systems, and implementation languages.
- It allows legacy systems to be "wrapped" in standard interfaces, giving them the ability to become distributed components as well.
- It allows the co-existence of solutions employing multiple languages.
- It protects the existing legacy systems, and yet future proofs what you develop today against language, platform and communications obsolescence.

Vendors of proprietary products are typically under pressure to differentiate their product with extensions. These value-added features can often lead to incompatibility and confusion about portability and interoperability. Vendors must balance standards compliance with maintaining an edge. This is not an easy task: a product that is completely standards compliant can be more easily replaced. To protect market share, middleware vendors must selectively add proprietary new features along with standard ones to maintain a hold (sometimes termed a compelling value proposition) on their client-base.

Free software is free from these pressures. Often, as second-generation products, their value proposition is that they support the standards (where they exist), very closely. This is their way to differentiate themselves from the other products, already present in the market place. They hope to use the consensus achieved in that standards community, and the experiences derived from multiple product implementations by users, as a way to stabilize the technology domain. Then from that technology create a commodity item and thus another stable building block in the technology layers that make up distributed systems. Users and vendors can then move on to other areas, along the value chain, higher up the ladder of abstraction, to advance technology and create new value.

Applications are the true value-added software. Middleware should be "low impedance" software for enabling application interoperability and systems diversity. Software historically has moved towards a monopoly, as a way to achieve easy interoperability, via uniformity. Within the middleware market there should be enough diversity to foster innovation and yet with sufficient uniformity to enable cooperation. Free software can act as a break in the natural progression towards a single dominant vendor. This progression towards a monopoly is not healthy for systems. It is termed pejoratively, "monoculture", because of its vulnerability to attacks by viruses etc.

In free software the users contribute capabilities to the product, which they want to have available. This might include extensions beyond the standard but they are created by a user's need, not by a vendor's motivation to lock in. These activities can become the basis for a standards submission as they can prove the viability of an approach and enable many users to validate their utility in real life situations.

Conclusion

In the past, standards emerged towards the end of product and technology cycles. Often they were too late to do any

real good: they really just codified past technologies. Newer standards, that try to get out in front, often lack the practical experience that really enables them to gain traction. Free software can offer the best of both worlds. A free software project can be a work-in-progress, upon which standards can be based, practically, and gives no feeling that the software is locking you in.

Early APIs can coexist with later ones, thus users can migrate over time, at their pace.

Openness is as much about supporting diversity, as it is anything else. Middleware is a powerful leverage point for ensuring diversity can be accommodated.

Linux on the desktop is part of, but by no means the whole story, for free software.

If you're not convinced, think on this: being an active contributor with a free software project offers you visibility, sharpens your skills, looks good on your resume, and enables you to participate in a virtual world-class team. It can give your present job an added edge.

Copyright information

© 2005 by Malcolm D. Spence

Verbatim copying and distribution of this entire article is

permitted in any medium without royalty provided this notice is preserved.

About the author

Malcolm Spence has a Diploma in Aeronautical Engineering from Kingston upon Hull College of Technology, did post graduate study at the College of Aeronautics, Cranfield, (both in England), and then obtained his MSC in Civil Engineering (structures) at UMR. He also attended jet engine school at Rolls Royce (UK) and the Tuck Executive Management course at Dartmouth. Malcolm has directed the free software line of business at Object Computing for the last five years. Prior to that, for a year, he was a freelance marketing consultant with various start-up companies. From 1981 to 1998 he was with St Louis sales office, and then the industry marketing group, at Digital Equipment Corporation in Boston. His later assignments involved providing marketing support for DEC Central Engineering OO initiatives in the Asia Pacific region. Before joining DEC, for thirteen years, he was an aeronautical engineer with the McDonnell Aircraft Division of McDonnell Douglas in St. Louis. He resides in the St. Louis area with his wife and three children.



A PLANT NEEDS WATER TO GROW

THE OPEN VOICE NEEDS SUBSCRIBERS TO PROSPER!

Free Software MAGAZINE

BY SUBSCRIBING YOU WILL BE SUPPORTING A MAGAZINE WHICH BELIEVES IN FREE SOFTWARE. ALL OUR ARTICLES ARE RELEASED UNDER THE GNU FREE DOCUMENTATION LICENSE, ENHANCING EXISTING INFORMATION ON FREE SOFTWARE.

SUBSCRIBE NOW!

[HTTP://WWW.FREESOFTWAREMAGAZINE.COM/SUBSCRIBE](http://WWW.FREESOFTWAREMAGAZINE.COM/SUBSCRIBE)

Smarter password management

How to handle your passwords without getting lost

John Locke

Y

our dog's name... your anniversary... your childrens' initials, birthday, or birth weight... your favorite hobby, or the name of your boat. Which one do you use for your password?

Network Administrators and hackers know that most people choose passwords like these to protect anything from logging into web-based bulletin boards to buying things online.

Why does it matter? Identity theft... corporate espionage... loss of your data, or digital images. Do you want to risk these things? In many cases, a weak password is all that separates your data from anyone who wants to impersonate you online, or worse.

The problem with weak passwords

Passwords that are simply names of pets, names of children, common names of any type, are called "weak passwords." Basically any word you can find in a dictionary or list of names makes for a weak password.

I don't like to use fear to motivate people, but practicing safe password management is as important as locking your house when you leave. Only whenever you're connected to the internet, it's like having a house in the worst neighborhood in the biggest city around and if you don't put a good lock on the door, you will get broken into, even if you're home.

The problem with strong passwords

If you work at a large company, they may not allow you to have a simple password based on any word you can find in

a dictionary. E-Commerce sites that have good security require passwords at least 8 characters long. They group the characters you type into four groups: capital letters, lower-case letters, numbers, and symbols, and then require you to have at least three out of the four groups represented in your password. And then they make you change your password every two or three months. This type of password is called a strong password.

Practicing safe password management
is as important as locking your house
when you leave

The problem is that you soon end up with many more passwords than you can possibly keep track of. You either forget your new password, requiring the administrator to reset it for you, or you start writing them down. Far too many people have their current passwords scribbled on a yellow sticky note attached to their monitor where anyone can see it.

With weak passwords, all an attacker needs to do to obtain them is go through your trash, or engage you in innocent conversation. With strong passwords, all he needs to do is visit your office. In either case, the attacker is engaging in a type of attack called *Social Engineering*, which is the easiest way to break into a system.

Do I always need a strong password?

No. Strong passwords provide far more protection against different types of attacks, especially those considered *Brute*

Force attacks. An example is something called a *Dictionary Attack*, where the attacker takes a list of words, sometimes an entire dictionary, and uses a special cracking program to try each word on your account. The dictionary used includes common animal and people names.

Many systems defeat these types of attacks by locking you out after a few failed attempts. But the real concern is what an attacker can do once they break into any particular system.

Assess your risks

There are low risk, and high risk computer systems. To avoid having 30 different passwords to remember, you can group together systems that have the same level of risk, and reuse your passwords. Many security experts would argue that this approach reduces security, but let's be realistic here: if you don't remember the password for a particular system, and then type in all of your "standard" passwords to try to log into it, you may have just compromised all of the systems that use any of those passwords.

There are many ways of grouping systems, but here's what I recommend:

Low risk systems

If you never give your credit card, drivers license, social security number, or any other sensitive information to a web site, you probably don't need to use a strong password. For sites like the New York Times, online bulletin boards, and the myriad of places that ask you to create an account before allowing you to post, use a throw-away, easy-to-remember password. The worst an attacker could do is impersonate you on a web site, a mild form of harassment, but nothing more serious than that.

You should realize that any time you type a password into a system that doesn't immediately take you to an encrypted site, your password could be intercepted by all kinds of unknown people. Look for the lock or key icon in your browser's status bar, and "https" in the web address. If these things don't appear, or there's a warning, don't trust the site. Use a weak password, and consider it public. As long as you trust a site as being legitimate, I consider it fine to reuse the same weak password for all of these types of sites.

Medium risk systems

You might not agree, but I consider credit card information to be medium risk. To purchase things using a credit card at all, you have to take some risk: the waiter at the restaurant could copy your card when taking your payment; somebody

could eavesdrop on your cordless phone when you give the number to the pizza delivery place; or somebody could look over your shoulder in line at a store.

Credit Card companies provide you with protection here - you're usually only liable for the first \$50 of any misuse of your credit card. For many credit cards, the bank takes full risk for online payments. You have to report charges you did not make in writing within 60 days, and these guarantees don't apply to debit cards, but overall loss of your credit card amounts to a bigger hassle but not devastation to your identity. So I recommend grouping all web sites you use a credit card for into a "medium risk" group. If you give a web site a credit card, you're already trusting them to not make bogus charges so you can probably trust them to not try to use your strong password on other sites.

A weak password is all that separates
your data from anyone who wants to
impersonate you online, or worse

Some cautions here:

- Never send a credit card number, or any more sensitive information, through an email system that is not encrypted. If your email system is encrypted, you'll know it: you'll have to do quite a bit of work on both the sending and receiving end, so assume your mail is completely insecure, because it is.
- Always make sure the web site is encrypted before typing in your password. Look for the lock or key icon in your browser window. In Firefox, the address bar (where you type the web address) will turn yellow if it's properly encrypted.
- Never use a public computer to make web transactions. Even if the web site is encrypted, there could be snooping software installed on the computer that could get your user account and password as you type it. Only conduct sensitive transactions on computers you trust and get the spyware off first!
- Just because a web site is encrypted, doesn't mean your data is protected. Many smaller companies have not invested in proper security to protect your password and credit card information. If in doubt, look for a security statement, or ask! If your business would like to properly secure customer data, contact **Freelock Computing** (<http://freelock.com/mail.php>) and let's talk!

High risk systems

Any system that contains your social security number, drivers license number, or other financial account numbers should be considered high risk. Systems that contain sensitive business information should be protected with a strong password, and if they're connected to the internet, that password should be changed frequently.

As a general rule, never give your password to anyone, especially not a password you use in other medium or high-risk systems

For the most part, this means treating your laptop or workstation as a high-risk system so use a different password to log into it than you use for e-commerce or general use.

In most cases, you can get by with three passwords, using them on the appropriate level of system: a weak password for general, low risk systems; a strong password for e-commerce and medium risk systems, and a different strong password for any computer you use that has business or sensitive information on it. In some cases, this isn't enough. If you have critical systems that contain personally identifiable customer data, or administrative access on customer machines, you may need to manage dozens of passwords. We'll cover how to securely manage dozens of passwords, as well as create strong ones, next month.

As a general rule, never give your password to anyone, especially not a password you use in other medium or high-risk systems. If you're getting help from somebody who administers a service for you, they will be able to set your password to something else without knowing your password.

Copyright information

© 2005 by John Locke

Verbatim copying and distribution of this entire article is permitted in any medium without royalty provided this notice is preserved.

About the author

John Locke is the author of the book *Open Source Solutions for Small Business Problems*. He provides technology strategy and open source implementations for small and growing businesses in the Pacific Northwest through his business, **Freelock LLC** (<http://www.freelock.com>).

A PLANT NEEDS WATER TO GROW

Free Software MAGAZINE

BY SUBSCRIBING YOU WILL BE SUPPORTING A MAGAZINE WHICH BELIEVES IN FREE SOFTWARE. ALL OUR ARTICLES ARE RELEASED UNDER THE GNU FREE DOCUMENTATION LICENSE, ENHANCING EXISTING INFORMATION ON FREE SOFTWARE.

SUBSCRIBE NOW!

WWW.FREESOFTWAREMAGAZINE.COM/SUBSCRIBE

THE OPEN VOICE NEEDS SUBSCRIBERS TO PROSPER!



ATTENTION: SENIOR AND IT EXECUTIVES. GAIN FREEDOM AND SAVE THOUSANDS OF DOLLARS!

ELIMINATE PROPRIETARY SOLUTIONS BY IMPLEMENTING AN OPEN SOURCE SOFTWARE STRATEGY!

RECEIVE A **FREE** OPEN SOURCE KNOWLEDGE STARTER PACK CONTAINING EVERYTHING YOU NEED TO KNOW ABOUT OPEN SOURCE SOFTWARE, ITS BENEFITS AND HOW TO MAKE IT WORK IN YOUR BUSINESS.

CONTACT US **BEFORE 1 APRIL 2005** AND ALSO RECEIVE A **FREE** OPEN SOURCE ASSESSMENT AND A CONSULTATION SESSION WORTH **\$500** FROM YOCTO'S BEST OPEN SOURCE SOFTWARE GURU (GAUTENG, RSA ONLY) WITH NO OBLIGATION AT ALL.

YOCTO OSS HAS THE KNOWLEDGE, EXPERIENCE, AND THE SKILLS YOU REQUIRE TO IMPLEMENT SUCH A STRATEGY. YOCTO OSS PARTNERS WITH MAJOR PLAYERS IN OPEN SOURCE SOFTWARE AND IS ACTIVELY INVOLVED IN THE OPEN SOURCE SOFTWARE RESEARCH COMMUNITY.

CONTACT US VIA OUR WEBSITE [HTTP://WWW.YOCTO.CO.ZA](http://www.yocto.co.za) OR EMAIL MARKETING@YOCTO.CO.ZA WITH YOUR CONTACT DETAILS AND THE REFERENCE NUMBER FSM001 TO RECEIVE YOUR FREE OPEN SOURCE KNOWLEDGE STARTER PACK.

*Make use of open source software to fully support your business strategy.
Do it without any risks to your business!*



The content tail wags the IT dog

Without hardware and software, there would be nothing for digital media to be created on, or used with. And yet the content industry attempts to tell the far larger IT industry what it can and cannot do.

Daniel James

The content industries have conspicuously failed to create a business model based on paid content over public IP networks, but still cling to the idea that those networks were created for just that use. Any software or system which might interfere with this theoretical paid content business is considered not just heretical, but probably criminal. The music and movie consortia have turned the transition to network distribution into a “with us or against us” battleground, with most of their customers fighting for the wrong side.

RIAA, copyright and file sharing

In a networked world, we would be lost if we couldn't find data on other computers. But four college students in the US have recently been made painfully aware that the mere act of searching and indexing networks can now be considered ‘contributory copyright infringement’. Without warning, they were sued by the Recording Industry Association of America for damages of up to US\$150,000 per song discovered on the campus LAN. Their crime was to have hosted search tools which indexed the files made available to guest users by fellow students - whether those files were music, or college coursework.

Of course, they weren't the only people running search engines on their respective campuses - any copy of Microsoft Windows includes SMB search tools which can perform the same function. The prosecution of these four individuals seems designed to create an example; to let the general public know that the content industry will go to any lengths, no matter how vindictive, unpopular or impractical, to defend the paid content dream.

Faced with a potentially bankrupting legal action, the stu-

dents settled out of court for between US\$12,000 and \$17,000 each. Some of the students have asserted that they still don't believe they have done anything wrong, while being constrained by the terms of the settlements as to what they can say in public. Although these sums are tiny compared to the original damages claims and would hardly cover the RIAA's legal fees, they represent the college savings of these four people. The RIAA has indicated that it will be pursuing more individuals for damages, and that the sums it is prepared to settle for will only get higher.

Never mind that an index of SMB shares on a particular LAN is an equally useful aid to institutions attempting to discover copyright violations as it is to the freshman who wants to hear the new Madonna album before he buys it. Never mind that the RIAA has exploited technological ignorance in its attempt to equivocate these college search engines with Napster, which was supposed to be a profit making company.

By extracting multi-thousand dollar tributes from individuals who haven't made a penny from the sharing of copyrighted work, the RIAA has blown apart its rationale for legal action - that of the artist as the victim. It provides evidence that the content industry really does feel more threatened by new forms of network distribution than it does by illegal counterfeit CD and DVD pressing plants. We haven't yet seen individuals given punitive fines for advertising car boot sales, or signposting street markets where counterfeit merchandise might be available.

Where are the thieves?

Where might the principle of “contributory infringement” end? Once again, as in the deCSS cases brought by Holly-

Textbox 1: Get your money for nothing...

Faced with a failing business model? Can't keep up with technology, or shift the units like you used to? If you've got nothing left to lose, why not try the secondhand ideas business?

The SCO Group (the company formerly known as Caldera, not the original proprietary-UNIX-on-Intel Santa Cruz Operation) recently compared Linux developers and users with peer-to-peer music sharers. In the infamous "Letter to Linux Customers" - not that SCO ever had very many Linux customers - CEO Darl McBride wrote: "Similar to analogous efforts underway in the music industry, we are prepared to take all actions necessary to stop the ongoing violation of our intellectual property or other rights."

In fact, SCO/Caldera was doing much worse business than most of the music industry until it decided to make source code licensing a mainstay of its revenue stream. The company was losing millions of dollars each quarter like the dot-com operation it is - until the SCOSource initiative was launched. Now it has reported a net income of US\$4.5 million for the quarter to April 30, 2003 on revenue of \$21.4 million. SCOSource was said to be responsible for \$8.3 million of that revenue.

This model of licensing inherited 'intellectual property' is also a potential saviour for the recording industry. If actually producing and distributing new music is no longer profitable enough to interest the major labels, they could simply re-hash old material - by starting a 1980's revival, for instance. In this analogy, the SCO source code licensed from Novell would be the back catalogue of Dire Straits; very popular in its day, but few people would admit to keeping a copy around now.

Having first made sure that the copyright signed over by artists was extended for long enough to extend past the furthest profit forecasts - say, the artist's death plus seventy years - the same old rubbish could be recycled for each new generation of music consumers. Why hasn't the music industry thought of this already?

wood, exercising the right to create a link from one machine to another across a network has suffered from legal intimidation. An open IT system has the potential for as many abuses of copyright as it has for creativity, not just within the operating system but from the hardware to the network and the unfettered blank media. But we probably won't see the RIAA take the IT companies, telcos and consumer electronics manufacturers who make file sharing possible to court.

In the essay Content is Not King, Andrew Odlyzko pointed out that only a minority of the value of public networks consisted of copyrighted, professionally produced 'content'. If we attempt to illustrate this point by comparing annual revenues across the industries, we discover there are probably several IT companies which each have greater annual revenue than the entire global music industry.

According to the International Federation of the Phonographic Industry, the international counterpart to the RIAA, recorded music sales worldwide were down 7% to US\$32 billion in 2002. Gartner estimates that the global expenditure on IT products and services in the same year was also down, which might suggest economic conditions rather than nefarious students as the cause of music industry woes. Including telecommunications, the figure for global IT spend in 2002 was \$1.521 trillion, of which \$556 billion was for

IT services alone.

The global media companies which make up the content industry have a well organised publicity and lobbying machine, which has so far been able to lead the public debate on how media and IT networks should converge. Accordingly, people who listen to music on computers are thieves, while those who listen to the same music on radios are true fans. That there is less and less difference between computers and radios seems not to matter. The spin on IT perpetuated by the content industry has also lead to statements that every CD-R sold represents a lost album sale. Have these people not heard of data backups?

Striking back

Every time a music or film industry spokesperson appears in the media to call for tighter controls on technology, there should be a rapid rebuttal from a technologist. When the vested interests of the content industry claim to be defending the artist in front of a political committee, there should be someone there sticking up for the long-established rights of everyone else.

The Soviet Union managed to control the creative technologies to the extent that dissidents had to make "samizdat"

copies one at a time, by hand - turning the distribution of ideas back to what it had been before Gutenberg. Given the opportunity, the content industry would love to limit powerful and versatile general-purpose computing to selected and approved individuals who could be trusted to respect the rules laid down by the owners of culture. It may seem an extreme comparison, but both the Soviet regime and the content industry have attempted to outlaw unencumbered copying machines.

It's up to the IT industry to educate the public and politicians of the real value of open systems. The music business is small compared to the industry it seeks to dominate. And yet, if it can get enough political support, it may end up doing just that.

Bibliography

- [1] Content is Not King (http://firstmonday.org/issues/issue6_2/odlyzko/)
- [2] Global sales of recorded music down 7% in 2002 (<http://www.ifpi.org/site-content/statistics/worldsales.html>)
- [3] 2002 Worldwide Hardware and Software Product Support Market Size and Forecast (<http://www4.gartner.com/DisplayDocument?id=383387>)
- [4] An analysis of the RIAA's complaint against Daniel Peng (<http://barillari.org/papers/peng/peng.html>)
- [5] Daniel Peng's web site (<http://m-net.arbornet.org/~danpeng/>)
- [6] Jesse Jordan's web site (<http://www.chewplastic.com/>)

Copyright information

© 2005 by Daniel James

(The following license is effective immediately)

You may not reproduce or retransmit the materials, in whole or in part, in any manner, without the prior written consent of the author.

About the author

Daniel James is a freelance technology writer.

Looking for affordable web solutions?

Website Builders

Shopping Cart Software/Programs

PHP Software

Ecommerce Solutions

GraFX SOFTWARE SOLUTIONS

www.grafxsoftware.com

Motivation and value of free resources

Wikipedia and PlanetMath show the way

Aaron E. Klemm

In October of 2000, web-savvy math students lost a critical education tool. MathWorld, an online encyclopedia of mathematics, vanished from the web leaving students, educators, and mathematicians with only a notice that legal problems had caused the shutdown.

MathWorld was an early example of useful web sites for education. Eric Weisstein, the author, originally started it as “Eric’s Treasure Trove of Mathematics.” He spent years collecting and writing entries for what would eventually become a highly regarded reference encyclopedia. As the site became increasingly popular, he struck a deal with CRC Press to publish a print version of his work.

Weisstein had accepted a position at Wolfram Research, and the company offered to help enhance MathWorld and provide hosting for the site. Meanwhile, conflicts with CRC Press began to surface. They wanted to disable portions of MathWorld in order to promote sales of the print version. CRC Press eventually used their contract with Weisstein to claim rights over large portions of his work.

A replacement emerges

As the MathWorld lawsuit dragged on, several students at Virginia Tech and others from IRC math channels launched PlanetMath, a web site to replace the type of resource Weisstein had created. From the outset they aimed to create a collaborative, community-driven site and chose the GNU Free Documentation License (GFDL) to cover the articles and contributions. The GFDL allows anyone to freely redistribute PlanetMath articles.

“We were all in an essentially defensive mood at the time, after what happened with MathWorld,” Aaron Krowne, a

principal of the project said. “We wanted to ensure that no third party could come along and ‘steal’ the PlanetMath content,” he continues. The GFDL allowed them to do that and guaranteed to contributors that the PlanetMath staff would not unfairly profit from their work. The license allows the authors to retain rights to their own contributions.

Fig. 1: A lawsuit shut down MathWorld



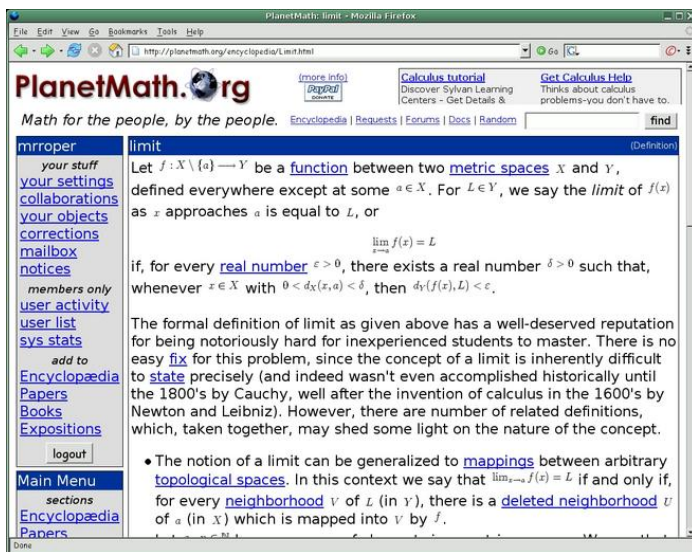
Interestingly, there are plans to create a print version of PlanetMath as well. The GFDL ensures PlanetMath will not encounter the problems MathWorld did.

Krowne was heavily influenced by Yochai Benkler’s paper,

“Coase’s Penguin, or Linux and the Nature of the Firm”. Benkler’s thesis challenges economist Ronald Coase’s belief that production is most efficient in firms and markets.

“I realized that the conditions and attributes that make free software so great are actually just a special case of something Benkler calls ‘Commons-based Peer Production’, or CBPP,” Krowne says. He lists PlanetMath along with GNU/Linux and Wikipedia as examples of what CBPP can achieve.

Fig. 2: PlanetMath is a collaborative mathematical encyclopedia



Benkler writes, “Removing property and contract as the organizing principles of collaboration substantially reduces transaction costs involved in allowing these large clusters of potential contributors to review and select which resources to work on, for which projects, and with which collaborators.”

Since its inception, more than 7000 users have registered on PlanetMath and the encyclopedia section has grown to nearly 4000 entries. Krowne summarizes the benefits of peer-production, “I think a great amount is gained by blurring the line between producer and consumer of content.”

How much can users produce?

Wikipedia is blurring the lines of production with astounding success. Edited entirely by volunteers, the collaborative online encyclopedia has grown to over one million articles with versions in more than 40 languages.

Founders Jimmy Wales and Larry Sanger originally started Nupedia, a traditional encyclopedia with expert authors and strict review standards to ensure article quality. Nupedia

was innovative only in that it was published on the web for readers at no charge.

After about a year of work, just twenty-four articles were complete and funding was drying up. During that time, Sanger discovered Wiki technology that allows collaborative document editing of web sites. He implemented it to enhance the production of articles prior to submission into Nupedia’s extensive review process. He named the setup Wikipedia and all articles were placed under the GFDL.

As Nupedia fizzled, the developers turned their focus to Wikipedia. The project quickly drew thousands of contributors eager to write about their areas of expertise. Wikipedia now has more entries and is published in more languages than any encyclopedia ever produced.

Bandwidth is expensive

Wikipedia relies heavily on donations to fund the expensive infrastructure necessary to keep it operating. Three primary costs in creating a project like Wikipedia are development, marketing, and distribution.

Wikipedia and PlanetMath have solved the development and marketing costs by leveraging their unique organizational structure. Contributors are motivated in many ways to help improve the projects: sometimes for recognition, sometimes out of gratitude, sometimes for the challenge, and usually with a commitment to the community they are building. The work gets produced and the proof is visible for all to edit.

Despite the lack of marketing budgets, these projects draw wide interest and are well known on the web. Wikipedia, for example, is now among the top 300 most-visited web sites according to Alexa traffic rankings. Their altruistic and novel nature also generates interest from the press and serves as a no-cost marketing tool.

They wanted to disable portions of MathWorld in order to promote sales of the print version

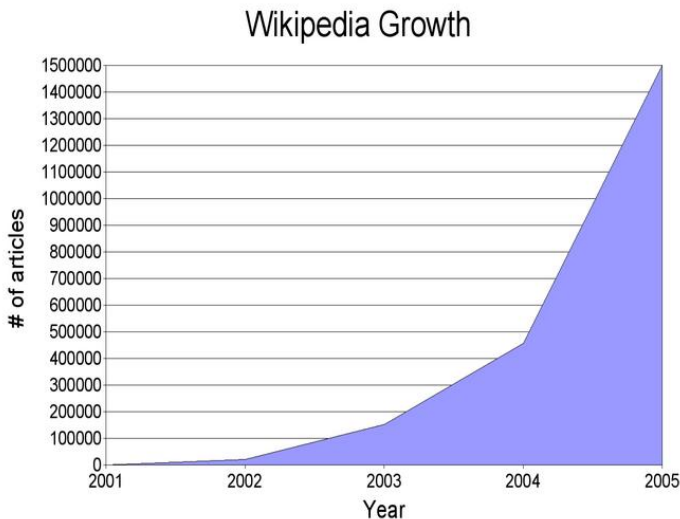
Product distribution can still be problematic for both projects, as servers and bandwidth invariably require time and money to keep serving pages. Wikipedia and PlanetMath continue to face challenges in maintaining a strong infrastructure. It is a consequence anyone could predict.

Developers and advocates of Free Software, Open Source, and peer-produced projects are often questioned about their profit motives. Critics are reflexively suspicious when they

focus on the no-cost aspect. They wonder about quality, support, and longevity.

This author, when explaining to a co-worker his insistence on finding a freely-licensed font, was asked, “How do you expect the font developers to get paid?” A programmer friend suggested this response: “Are you a font developer? Of course not, so don’t worry about it.”

Fig. 3: The number of Wikipedia articles each year since 2001



It was tempting to fire back with that response, but the witty retort would have been a day late, and the co-worker’s genuine concerns about sustainability would have gone unanswered. What the co-worker missed, of course, was that these projects are free of restrictions as well as cost. Relinquishing exclusive, restrictive ownership invites creative minds to extend the ideas implemented in a given project.

“I think a great amount is gained by blurring the line between producer and consumer of content”

Without generating funds directly from the product being produced, free sources (a phrase used for the remainder of this article to mean Free Software, Open Source software, or textual works licensed for freedom of re-use) rely on altruistic contributions, donations, and increasingly more funding from commercial companies that have some interest in further improvement of the project. GNU/Linux has been a significant example of the latter funding model.

Economically, the value opportunities created by Wikipedia are directed outward rather than inward. Third-parties are

given the unique opportunity to profit from works they did not develop. These uses of the Wikipedia database are becoming more visible with interesting results

I’ll take that, thank you

At first glance it may appear that third-party use is going the way a critic of free sources might expect — leeching. Thefreedictionary.com, by Farlex, Inc., mirrors Wikipedia’s database and places advertisements into the articles as a way to generate revenue. This kind of use is allowed under the GFDL, and Farlex does not technically violate the licensing terms.

The spirit of the license may be under attack in this case, however. Farlex attempts to manipulate search engine results in order to rank their copies of Wikipedia articles higher than those on Wikipedia itself. Farlex relies on this result to increase the chances of site visitors clicking on their ads.

When a user lands on Farlex’s site rather than Wikipedia, several things happen that do a disservice to Wikipedia. First, Farlex does not include an option to edit the page thereby eliminating the possibility for readers to improve the original text. Second, the information Farlex uses is necessarily older than the up-to-the-minute text on Wikipedia. Finally, while GFDL notices are in place on Farlex’s site, the sense of community development is stripped from the text.

Their altruistic and novel nature also generates interest from the press and serves as a no-cost marketing tool

It is a consume-only proposition for site visitors and a “click-our-ads” proposition from Farlex. Wikipedia users are not overly worried about the problem, but they have proposed a variety of solutions. If Wikipedia’s own entries continue to be the most sought-after results, the way linking works on the web may prevent lower-quality links, such as those to Farlex’s site, from jumping ahead in search results. A Wikipedia user concerned about the clone problem recently wrote, “As Wikipedia content proliferates, Google users are going to get more and more annoyed when they do a search and find 15 URLs of cloned material in the top 30 results. As a result, Google will have no choice but to fix this problem eventually.”

As Wikipedia contributors debate that problem, Farlex continues to gain — possibly at the expense of Wikipedia. Is leeching the best free sources can expect from third-party users?

Free helps free

Programmers are thinking of newer and better ways to integrate web resources into desktop applications, their own web applications, and other software. A software program known as Beagle, from the developers of the freely-licensed Gnome desktop, takes the concept of aggregating disparate resources to a new level. Beagle is not yet officially released with the Gnome desktop, but the code is under very active development.

Beagle tracks a computer user's activities in applications such as email and instant messaging and attempts to derive — from keywords, file types, and other clues — the context of this activity. Then it scans old email, discussions, and documents, which it displays on the desktop for easy and timely retrieval.

For example, while a user drafts an email to a friend about a great new band, Beagle will pull up a list of music files by the band, previous emails that mention the band, and perhaps search results from Amazon about the band's CDs.

In order to make the extension from email archives and the file system across the internet to Amazon and perhaps Wikipedia, Beagle developers must first determine which web resources will allow that kind of use. The Beagle developers have had good luck with Amazon because the company actively promotes web services for external use of their data.

There are plans for Beagle to support encyclopedia lookup features in the future. The developers can be sure licensing problems will not stop them if Beagle starts allowing contextual searches of Wikipedia's vast repository of GFDL articles.

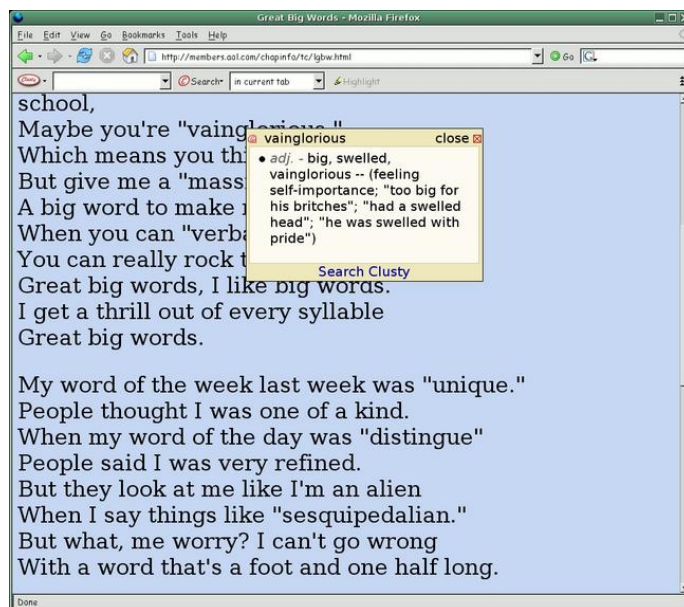
If Beagle has a profound influence on the attractiveness of Gnome (attractive Gnomes?), each third-party currently profiting from the free desktop system will receive a boost. Red Hat and Sun, who have already have successful profit strategies around Gnome, will indirectly benefit from Wikipedia. It will make their products an easier sell.

A responsible hybrid

The Clusty search engine from Vivisimo, Inc. uses Wikipedia articles directly in their system. Clusty searches

various data on the web and creates “clusters” of links to help organize query results. They mirror a complete copy of Wikipedia for this purpose.

Fig. 4: Clusty's Firefox toolbar enhances page viewing.



“Like the other information sources available on Clusty, Vivisimo's clustering technology helps users find the desired results while discovering unexpected relationships,” Marco Arment, a software engineer at Vivisimo, said.

As an example, Arment explained that a search for “natural language processing” on Clusty will bring up the related articles with links to Wikipedia articles as well as related, clustered results. In one of the clusters is a link to the Association of Computational Linguistics which is something that does not appear in the Wikipedia article.

When a third-party improves upon the work of a free source, that innovation can be re-implemented into the original work. Arment believes third-party users often become contributors to the projects they borrow from and help in other ways. “Third-party use can also highlight new uses for the resources that the creators may not have considered, leading to future expansion and improvement,” Arment said.

Clusty provides a custom toolbar for the Firefox web browser. The toolbar interfaces with Clusty and allows right-clicking on any word in a given web page to access encyclopedia and dictionary “clips” related to the word. A definition or summary of encyclopedia entries pops up in a small window containing contextual links for that word.

Arment thinks free sources provide Vivisimo with unique opportunities to innovate. “In addition to cost savings, free sources give us greater flexibility to use the data in innovative ways. If Wikipedia had a restrictive license, for ex-

ample, licensing conditions and limitations may prevent us from presenting Clusty Clips with the Firefox toolbar,” he said.

“We’re providing additional value to the Wikipedia content by increasing its availability, and we have countless ideas for future expansion of this concept,” Arment said.

Value, value everywhere

Despite the critics and because of their structure, free sources continue producing quality work. Value opportunities crop up around Wikipedia with no extra effort required by Wikipedia. All players are invited to explore the exploitation of these opportunities.

Because the economic value creation of these projects is not directed inward toward their own sustenance, should we worry that these projects will degrade? If that happens, the value third-parties rely on will degrade along with the project. Responsible third-party players will not be able to ignore this incentive to help sustain free sources.

As each new project proves it can at least exist — a low bar that some critics have seen as too high for the short legs of free sources — third-parties with different goals gain a stake in sustainability. They become part of the community in the

same way contributors and users do.

As Marco Arment puts it, “Every third-party use of a free resource also lends credibility to both the resource itself and the ‘free’ concept in general.”

Copyright information

© by Aaron E. Klemm

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.2 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license is available at <http://www.gnu.org/copyleft/fdl.html>

About the author

Aaron Klemm is an advocate for freedom of software and knowledge resources. He is co-founder of Mathforge.net.



Linux Based Solutions

www.snowdoniait.com

Tel: 01248 355 985 Mob: 07976 632 572

SNOWDONIA

IT SERVICES



By using our services you will gain the many benefits of Linux and free software:

- A secure, stable operating system
- A reduction in licensing costs
- A wide variety of applications for home and office use
- A powerful, customisable system

Services available include:

- Installation and configuration of server and desktop systems with appropriate applications
- Training in the use of your Linux system
- Website design with hosting on Linux servers
- Monitoring systems with customised plugins
- Vulnerability scanning and information security programme development

Contact us for a demonstration of Linux and the applications available, or to enquire about our Linux based solutions for home users and businesses.

On-site daytime and evening visits available. Competitive rates, free estimates.

It's all about freedom

Freedom is free software's competitive advantage

Christian Einfeldt

M

aybe it's true that a "Rose by any other name still smells as sweet," but not being able to easily pronounce the name of software is a big turn off to exploring it.

That's true whether the name of your word processing program is "Espronceda" or "Microsoft Word" or "OpenOffice.org Writer".

Now... unless you can read Spanish and are familiar with the digital literacy efforts of the local Regional government of Extremadura, Spain, you probably would not have recognized the name, "Espronceda" in the paragraph immediately above. You would have had absolutely no idea what that program did, and you probably thought that it wouldn't be written in a language you could understand, so why bother? My point exactly!

The ability to change and adapt is software libre's ace in the hole

The name "Espronceda," is the name that Extremadurans gave to their flavor of OpenOffice.org Writer. Extremadura is one of the seventeen Regions of Spain (The term "Region" in Spain is closely analogous to the term "Province" in Canada or "State" in the U.S.A.). Each Region has its own unique government, history, culture, and in many cases, its own language. The government of Extremadura has made digital literacy a prime policy priority. To speed the adoption of software libre in Extremadura, the officials there decided to rename OpenOffice.org Writer as Espronceda, after the famous Spanish poet, Jose de Espronceda.

It's a beefy issue

This issue is big... very big! The name given to a software application is what simple end users like me and my six billion closest friends first encounter, and it shapes our view of the code and helps us to answer the biggest questions that a simple end user asks when facing a new application for the first time: "Why should I use this code?", "How do I use this code?" and "Why should I care? Isn't a computer just like a toaster?"

The ability to change and adapt is software libre's ace in the hole. As a disruptive technology, software libre relies on disruptive distribution channels, such as the free telecentros in Sao Paulo, Brazil, or the public libraries in Scotland. These channels expose new users to an application without a sale occurring. Many of the users of the 120 Sao Paulo telecentros do not own computers themselves. Indeed, some of them don't even have a reliable electricity supply. The telecentros are located in the heart of the poorest favelas (slums) which ring Sao Paulo, a city of sixteen million souls. Likewise, the Extremaduran telecenters service typically underserved users who, for what ever reason, are too intimidated or poor to have a computer in their homes. Similarly, the Scottish libraries circulate copies of OpenOffice.org to those who cannot afford Microsoft Office.

This is a huge market we're talking about

Of those six billion closest friends to which I was referring above, only about a billion live in countries with average annual incomes in excess of US\$10,000.00. The remaining five billion live in countries well below that annual average. The ability of users in those countries to localize software

libre into their own languages represents a huge advantage for free software. Proprietary software companies have created the impression that it is improper to change the names of applications, and have locked down the code so that localization is not possible without the source code. This practice shifts the cost of localizing to these companies; whereas companies such as Sun and Novell will gain the benefit from the externalization of some of the localization costs to the shoulders of the local communities, who are the masters of their local dialects, anyway.

Proprietary software companies have created the impression that it is improper to change the names of applications, and have locked down the code so that localization is not possible without the source code

The local Extremaduran government was in a better position to choose the name of Espronceda for its version of Writer than was Sun Microsystems or the global OpenOffice.org community. Each local community is in the best position to understand what will make sense for its residents. Leaders in those communities best understand how to go about the process of winning “buy-in” from local users, and what potential names best reflect the local cultural proclivities for approaching a novel technology.

Penguinistas are familiar with the advantage of modular interfaces when it comes to fitting pieces of technology together, but it's also helpful to see the GUI itself as a key modular strength of software libre. This view might seem rather straightforward, but it's actually rather contentious and difficult to implement in practice. Michael Robertson, CEO of Linspire, recently had a conversation on the English language OpenOffice.org marketing list about this issue with Bruce Byfield, an independent columnist and contributor to the OpenOffice.org project, and author of a user manual for OpenOffice.org.

Bruce Byfield felt that while it was a good thing for Linspire to pursue widespread proliferation of the code (what Linspire humorously calls “flouridation”), mere flouridation without more deliberate education was not enough:

“ As for the comments about [Linspire] vendor lock-in, they refer specifically to Linspire's re-labelling of software packages for the purposes of branding, and - so far as I can see - an absence

of any mention of antecedents in its general advertising and presentation of packages. I know a number of non-geeks, for example, who are under the impression that Linspire is a completely new operating system, rather than an adaptation of GNU/Linux in general and Debian in particular. While this re-branding makes sense in commercial marketing, I observe that it does not play well in free software/open source communities, where credit is often the only reward for effort.” (Bruce Byfield, 2004/10/15, OOo Marketing list server, <http://marketing.openoffice.org/servlets/ReadMsg?list=dev&msgNo=17344>)

Michael Robertson replied by saying that the market of folks who care about the philosophy of their computer software has been taped out, and that the bulk of remaining potential customers are only interested in out-of-the-box functionality:

“You are right we do not emphasize the philosophy. I wouldn't say we emphasize the technology either. We emphasize the benefit to the end user/retailer/OEM and to all parties it revolves around economics. The vast majority of OEMs and retailers don't care about the philosophy. They only care if there's a chance for them to make money. If there's no chance then they don't carry the product, it's that simple. If you start talking to Walmart about philosophy you will quickly be escorted out of the office. And they are not unique among major distributors or retailers that make business decisions based on economics. It has nothing to do with whether they agree or disagree with the philosophy behind free software it's just that's not how they make decisions. So how do you win them over? You have to make it economically beneficial for them to carry your products.” (Michael Robertson, 2004/10/15, OOo Marketing list server, <http://marketing.openoffice.org/servlets/ReadMsg?list=dev&msgNo=17357>)

Then Michael Robertson wrote something, which reminded me of Espronceda:

“They [OEMs and retailers] decide if the product is appropriate for their consumers, it matters little what we say. If they turn on the screen and are bombarded with a foreign language like Mozilla, Gimp, Gnome, KDE, Gaim, K3B, Gnu, Evolution, etc. its a very short conversation. This will confuse my users. Not interested – thank you. Come back when it’s easier. *You have to frame terms in the way that they’ll understand.* So ‘K3B’ becomes ‘CD Burner’ and ‘Mozilla’ becomes ‘Web Browser’ and ‘Gaim’ becomes ‘Instant Messenger’. This isn’t to slight the developers (all the Abouts and stuff remain the same). It is designed to make it palatable for OEMs and distributors as well as end users. Where the name is intuitive we don’t touch it. OpenOffice.org is sufficiently straight forward so we don’t change the name for any of the pieces OpenOffice Calc, OpenOffice Writer all keep the same name.” (Michael Robertson, 2004/10/15. [Italic added]. <http://marketing.openoffice.org/servlets/ReadMsg?list=dev&msgNo=17357>)

There is a common thread between Linspire’s marketing efforts and the efforts of the Extremadurans to increase adoption of free software. Both efforts involve a bit of splash and flash. Linspire has its “Click-and-Run” button for easy download; and the Extremadurans have actually gone to the effort of creating a slick cartoon character called **Linextremix** (<http://www.linextremix.com/>) to interest kids in trying out their local version of Linux, which they call “LinEx.”

Of course, disagreements about naming free software packages have a long and complicated history. The debate about whether to call the operating system “GNU/Linux” or to reserve that term solely for the kernel has been around since Linus first opened his bedroom door to release his kernel.

It’s a cultural difference

But there is a solution to this on-going issue, which deserves a fresh look. Consider the difference in naming applications to be a *cultural* difference, and approach it the same way any cultural difference would be approached. There are very few sober-headed folks who would assert that American English is “better” than British English, or that Castilian Spanish was “better” than any of the other dialects of Spanish,

or that Mandarin Chinese is “better” than Cantonese Chinese. Nor would any enlightened person insist that a recent immigrant to the US from another country should adopt an English first name such as “Tom” or “Sue” because their native language name was unfamiliar. Names and languages in those contexts are viewed as cultural artifacts, and are respected as such.

If we view the naming of free software code base as much a cultural artifact as language, we would find it easier to understand that Bruce Byfield, Linspire, Richard Stallman, and the Extremadurans all have a common valid point: choosing a name counts. Big Time! It has been my experience that people involved in the free software conferences I have visited attempt to speak the foreign languages of the cultures they visit, even if only a word or two. The software libre community generally prides itself as being multi-cultural. If we look at Linspire’s act of renaming the open code it uses as a cultural accommodation just as much as the Extremaduran effort, we will understand that North Americans have a cultural bias that tends to see technology mostly as a mechanistic servant, and nothing more.

Contrast that view with the views of South Americans I encountered during filming at the FISL free software conference in Porto Alegre, Brazil, in Spring of 2004. They tended to use the “GNU/Linux” appellation much more widely when referring to the whole operating system, and tended to reserve the term “Linux” for the kernel only. Many of my free software friends in North America, by contrast, just refer to the whole package as “Linux,” with the understanding that GNU is every bit as important to the functioning of the OS as is the X Windowing system. Their preference was based on the philosophy of freedom that is embodied in the code perhaps even more so than the low cost of adoption, although the latter certainly played a role too. They saw their ability to control the code as an opportunity for hemispheric technological independence and cultural advancement through enhanced digital literacy. Richard Stallman’s emphasis on freedom spoke to them, whereas the North Americans we have interviewed often didn’t see what the fuss was about. One culture sometimes values what another culture takes for granted.

All marketing is a local phenomenon

To further develop an earlier thought, the flexibility to choose a name for an application is one of the key competitive advantages of software libre. Consider this passage from the *Seeing What’s Next*, a 2004 publication of the Har-

vard Business School Press by co-authors Clayton Christensen, Scott Anthony, and Erik Roth:

“Consider the difference between Microsoft Windows and the Linux operating system. Windows is a highly integrated, interdependent operating system. To optimize the operating system, application developers must conform their products to meet Microsoft’s interface requirements. Efforts to try to modify Windows to improve individual applications would be disastrous; any individual change would have literally thousands of unanticipated consequences and operating system problems. Linux works the other way, because its goal is to enable optimized applications. The Linux operating system itself is modular. As long as you follow the rules, you can modify it to optimize the performance of an application.” (*Seeing What’s Next*, p. 20)

The ability of local users to create the name for free software applications to their own familiar names is a key to the *optimization* of that code for local use. The local users do not have to conform to the program’s GUI; rather, the GUI is modular and can conform to their needs. This should provide software libre with a key marketing advantage in numerous smaller markets, which the disruptive software libre can march up-market.

The world is a very big place, with people who have quite divergent needs. Software libre has an economic advantage in serving the needs of those diverse cultures

The current market leader is faced then, with an innovator’s dilemma: does it fundamentally change its business model and open its market-dominating code, such that local users can tweak it the same way that software libre can be tweaked; or does it bear the cost of changing that code; or does it ignore these impecunious markets and risk that free software takes a firm root there? The market leader has no attractive options here, as all possible options entail greater costs with declining fiscal reward.

The world is a very big place, with people who have quite divergent needs. Software libre has an economic advantage

in serving the needs of those diverse cultures. Both the for-profit and non-profit institutions, which serve the needs of this diverse population, will have huge market advantages over their competitors. Bruce Byfield and his friends might like Debian. The pragmatic customers in North America might like a simple out-of-the-box solution like Linspire. Richard Stallman told us during his interview for our film that he likes the **Ututo-e distro** (<http://linux-cd.com.ar/ututo/>) from Argentina because it’s all free software. These differences reflect deeply held value structures, which although different, are not incompatible. Whether you like to distribute the code, and then teach freedom when you have the newbies’ attention, or whether you like to talk about freedom to get their attention, you are helping to show people how to love Tux and the GNU gnu.

Consider the difference in naming applications to be a *cultural* difference, and approach it the same way any cultural difference would be approached

Copyright information

© 2005 by Christian Einfeldt

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.2 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license is available at <http://www.gnu.org/copyleft/fdl.html>.

About the author

Christian Einfeldt is the producer of the upcoming film “the Digital Tipping Point: the culture of freedom in cyberspace”, to be released in September, 2005. For more info, go **here** (<http://www.digitaltippingpoint.com>)

The Commons

The Commons as an Idea - Ideas as a Commons

David M. Berry

The concept of the commons has a long heritage. The Romans distinguished between different categories of property, these were: Firstly, *res privatae*, which consisted of things capable of being possessed by an individual or family. The second, *res publicae*, which consisted of things built and set aside for public use by the state, such as public buildings and roads. The third, *res communes*, which consisted of natural things used by all, such as the air, water and wild animals. The commons, or *res communes*, has had an important social function in our society, it provides a shared space, a resource that is shared within a community, a network of ideas and concepts that are non-owned.

Ever since the rise of capitalism, people have been putting fences around the commons and declaring “this is mine”

Ever since the rise of capitalism, people have been putting fences around the commons and declaring “this is mine”. In England, the phrase “good fences make good neighbours” has become unreflexive and normalised. The form of individualised property ownership within modern society, mediated through market exchange, has gradually colonised more and more of our social world. Should it also colonise the realm of information?

The Information Society

In the Information Age, our lives are increasingly mediated through digital technology. Through computers, technical

Textbox 1:

He who receives an idea from me, receives instruction himself without lessening mine; as he who lights his taper at mine, receives light without darkening me. That ideas should freely spread from one to another over the globe, for the moral and mutual instruction of man, and improvement of his condition, seems to have been peculiarly and benevolently designed by nature, when she made them, like fire, expansible over all space, without lessening their density in any point, and like the air in which we breathe, move, and have our physical being, incapable of confinement or exclusive appropriation (Thomas Jefferson, 1813)

devices and countless databases, servers, and storage systems, information has grown in importance and value. But, as information itself has become more crucial to modern society, so too has the desire to profit from it. Indeed, information, when viewed as a potential form of profit, justifies new ways of legitimating its ownership as a property right. And, of course, information when viewed as property seems to require fences; virtual fences that can both identify it as being owned, and prevent others from taking it without paying.

With the emergence of capitalism, more and more of the *res communes* was enclosed and transferred to the realm of *res privatae*. In the Second Treatise on Government, John Locke argued that by “mixing” our labour with the commons we transform it into private property, if you farmed the land it would become property. However, there might be

arguments and disputes regarding ownership, perhaps others might take the land from you, contest your ownership or even ignore your claims altogether. To avoid this danger and prevent “war of all against all” he proposed a social contract that would give property legal tenure in civil society. Following Hobbes, he argued that a state would need to be formed that would guarantee private property through a monopolisation of the use of violence and act as a neutral arbiter between different factions. Locke, though, made an important but often forgotten restriction on this initial acquisition of property, called the “Lockean proviso”. This states that individuals have the right of acquisition only if “enough and as good [is] left in common for others”.

Another famous justification for the “rationality” of private property is given in Garrett Hardin’s paper called “The Tragedy of the Commons”. This metaphor illustrates how individual’s interest conflicts with the common interest. In this article the Commons is a shared plot of grassland used by all livestock farmers in a village. Each farmer keeps adding more livestock to graze on the Commons, because he does not experience a direct cost for doing so. After a few years, overgrazing destroys the commons, it becomes unusable and the village perishes. It is often used to argue in favour of private property and against theories that defend communal ownership of resources. This narrative has been extremely influential, indeed much political and social policy is informed by the logic of this framework. However, it ignores the fact that property itself relies on a commons – i.e. laws that everyone agrees to abide by. A property regime only holds together on the basis of this common belief and shared understanding. In any case, the tragedy is only possible if you simplify human action to a selfish “rational choice” actor, rather than see human activity as extremely complex and part of a wider social network of norms and values.

When individuals contribute to a shared project that creates new ideas or even provides an important social function it becomes increasingly valuable

When individuals contribute to a shared project that creates new ideas or even provides an important social function it becomes increasingly valuable. It is no surprise that the temptation to appropriate and resell the products of the commons can be overwhelming. The current neo-liberal trend toward the privatisation of energy and communication

services is another example of public goods being enclosed and transformed into private property. Market regimes and neo-liberalism survive off these privatisations; of physical goods, such as the transistor; of distribution networks, such as energy or water; or of services, such as the National Health Service. The commons, which once were considered the basis of the concept of the public, are privatised, and the values of common ownership and the public good are destroyed in exchange for market exchange and consumer choice. The relation between the public and the common is replaced by the power of private property and the market.

Digital Revolution

The digital revolution has facilitated widespread cultural participation and interaction that previously was not possible. At the same time, it has allowed the creation of new technologies, potentially limiting and controlling these forms of cultural participation and interaction. The “expression” of ideas and concepts, such as books and music, can be encoded into digital information so that it can be transferred through communications, databases and web pages. The production and distribution of this information is a key source of wealth in the digital age and creates a new set of conflicts over capital and property rights that concern the right to distribute and gain access to information. With these restrictions on the access and use of information there is a corresponding restriction on the use of ideas and concepts. What is distinctive about ideas? Unlike physical objects, concepts and ideas can be shared, copied and reused without diminishment. No matter how many people use and interpret a particular concept, nobody else’s use of that concept is surrendered or reduced. But through the use of intellectual property law – in the form of patents, trademarks and particularly copyright – concepts and ideas can be transformed into commodities that are privately regulated and owned. An artificial scarcity of concepts and ideas can then be established. Much money is to be made when creative flows of knowledge and ideas become scarce products or commodities that can be traded in the market place. And, increasingly, intellectual property law is providing corporations with vast accumulations of wealth.

This legal exclusion is being supported by technological means. To do so, corporations and governments are currently developing and configuring ever more closed disciplinary technologies. These technical devices act as electronic fences, regulating access to those that have paid, those that are approved of and those that consume. Digital rights management software, for example, sequesters and

locks creative works, preventing their copying, modification and reuse. Adobe e-Books, for example, can restrict to a fine level of granularity how you can use the text, the publisher can even mandate how many times you can print pages from the book, whether you can copy it, or if you can copy and paste sections into other texts. They can also set an expiry date for the book, so after a certain date the book will self-destruct and delete itself from the system.

Thus, public pathways for the free flow of concepts and ideas and the movement of creativity and the creative are being steadily eroded — the freedom to use and re-interpret creative work is being restricted through legally based but technologically enforced enclosures. Against this trend, a new global movement of networked groups that operate across a variety of creative media (e.g., music, art, design and software) is now emerging. These groups produce a gathering of concepts, ideas and art that exist outside the current property regime. The creative works of the Free/Libre and Open Source communities, for instance, can all be freely examined, challenged and modified. Here, knowledge and ideas are shared, contested and reinterpreted among the creative as a community of friends. The concepts and ideas of these groups, like the symbols and signs of language, are public and non-owned. Against the machinations of profit, these groups are in the process of constituting a real alternative.

Locking down Culture

Meanwhile, corporations are constructing the means to control ideas and concepts at a level of pay-per-view, whether watching, reading or listening. We all use and reuse ideas and concepts that are shared and non-owned without realising it. Changes are taking place due to the lobbying of the multinational media corporations and governments, particularly through the American use of TRIPs (Trade Related Intellectual Property agreements) and other international bodies such as the World Trade Organisation (WTO) — changes which are sadly lacking in democratic debate and deliberation. These moves threaten our ability to speak, write and even think differently (for if we can never read, see or hear concepts and ideas we can never use them).

An example of this new trend is given when Fox News Corporation trademarked the phrase “Fair and Balanced”. In August 2003, Fox sued the humorist Al Franken and his publisher E. P. Dutton/Penguin for alleging infringement on Fox’s three-word trademark “Fair and Balanced”. Franken’s book *Lies and the Lying Liars who Tell Them* was subtitled “A Fair and Balanced Look at the Right”. In the US, a dis-

trict judge refused to accede to Fox’s claim and Fox dropped the lawsuit but has retained the trademark. Next time they may be more successful as they and other multinational corporations lobby to strengthen the intellectual property laws when they are unsuccessful in court — for example, the Digital Millennium Copyright Act (DMCA) and the currently debated Induce Act.

The creation of new knowledge requires that ideas and concepts may be freely exchanged. If ideas and concepts can be digitally locked and controlled, it will have a devastating effect on our ability to draw on ideas from the past. A non-owned public domain, or commons, of freely shared concepts and ideas, where each may draw, without diminishing the availability of ideas and concepts for others is crucial. But whereas the enclosure of land contributed to the rise of capitalism and the power of the bourgeoisie to challenge the feudal order, this paper argues that the informational enclosure will conversely lead to a new feudal order. By drawing profit from the ownership of information the corporations will in effect be living from rents, a new rentier system based on the ownership of ideas.

This information-based system will allow the corporations — and they are predominantly corporations — who own the books and the newspapers, the music, the films, the patents and inventions to live off a monopoly rent from the rest of society. Taxing all members of society, maximising their profit and their income without any concomitant requirement to contribute creatively towards society. This movement threatens our ability as a society to re-use existing concepts and ideas and hence threatens social and cultural stagnation by closing our ability to be creative.

Our ability to use concepts and ideas is being restricted and controlled by an all encompassing and enveloping digital field that increasingly surrounds us

The corporations profit hugely from their libraries of art, films, music and writings, indeed, they need not worry about future creativity, as they increasingly own vast quantities of the creativity of the past. They can then package and resell this creativity in endlessly re-issued compilations, director’s cuts and special editions. As it is consumed it provides an endless stream of profit to the owners — for if you like it you’ll gladly pay again and again for the privilege of viewing. And should the founding ideals of intellectual property threaten profits — that copyright and patents should provide

a limited monopoly on ownership – the corporations lobby to extend the length of copyright terms. Indeed, corporations argue for unlimited ownership and control of creative works and new crimes to protect from the new “dangers” of informational theft, of so-called “piracy” and of “hacking”.

The Control Society?

Our ability to use concepts and ideas is being restricted and controlled by an all encompassing and enveloping digital field that increasingly surrounds us. Gilles Deleuze identified a control society, which moves beyond the disciplinary society that Michel Foucault observed. Rather than institutionally bound, such as in the school, the hospital or the prison, the control society monitors our every action. This form of digital surveillance is extremely well suited to observing and controlling our use of concepts and ideas, and will allow payment and punishment to be extracted in the use of any creative work. This new rentier world is being silently built around us, partly using existing legislation, such as copyrights and patents, but increasingly by the active construction of technologies of surveillance and control, digital rights management technologies (DRM), authentication and identity recognition systems.

To combat this threat, a new concept of the “commons” will have to emerge. New technologies of the commons will need to be developed. New stories will need to be told and new metaphors and common-meanings created. Rousseau said that the first person who wanted a piece of nature as his or her own exclusive possession and transformed it into private property was the person who invented evil. What is common, however, is good.

Copyright information

© 2005 by David M. Berry

This article is made available under the “Attribution-Share-alike” Creative Commons License 2.0 available from <http://creativecommons.org/licenses/by-sa/2.0/>.

About the author

David is a researcher at the University of Sussex, UK and a member of the research collective The Libre Society. He writes on issues surrounding intellectual property, immaterial labour, politics, open-source and copyleft. Further information can be found at <http://www.libresociety.org/>



A PLANT NEEDS WATER TO GROW

THE OPEN VOICE NEEDS SUBSCRIBERS TO PROSPER!

Free Software MAGAZINE

BY SUBSCRIBING YOU WILL BE SUPPORTING A MAGAZINE WHICH BELIEVES IN FREE SOFTWARE. ALL OUR ARTICLES ARE RELEASED UNDER THE GNU FREE DOCUMENTATION LICENSE, ENHANCING EXISTING INFORMATION ON FREE SOFTWARE.

SUBSCRIBE NOW!

[HTTP://WWW.FREESOFTWAREMAGAZINE.COM/SUBSCRIBE](http://WWW.FREESOFTWAREMAGAZINE.COM/SUBSCRIBE)

Are your systems **OPEN** for business? *Our customers' are!*

For over 10 years, **OCI** has been helping organizations build their mission critical infrastructure with open architectures using standards-based OO technologies.

OCI provides the following commercial-grade products to the **open-source** community:



ACE - The high performance C++ portability layer combining socket-level performance with type-safe programming, across a wide range of operating systems.

TAO - An **open-source** C++ CORBA 2.6 implementation with the widest range of services and span of platforms of any ORB.

JacORB - The **open-source** Java CORBA 2.4 implementation.

MPC - An **open-source** cross-platform build tool - "script once, build many"

OVATION - A tool for instrumenting & visualizing CORBA systems behavior.

Can't wait to go open? **OCI** offers a comprehensive set of migration support services that will help your organization minimize risk, cost, and time when moving to these **open-source** solutions.

To learn more about these products & services, visit: www.theaceorb.com

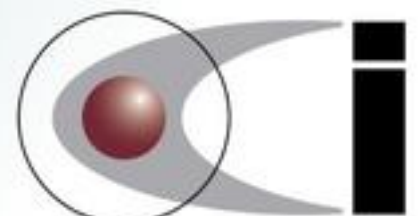
Product Development • Consulting • Educational Services

Object Computing, Inc. (OCI)

St. Louis, MO Headquarters: +1 (314) 579-0066

Phoenix, AZ Office: +1 (480) 752-0042

www.ociweb.com



OBJECT COMPUTING, INC.

Let's not forget our roots

Free software is not just about cost or stability: free software is a movement that mustn't forget the principles which made it possible

Tom Chance



GNU/Linux is growing all the time: new software is being created; new copies downloaded or bought; new users are discovering free software for the first time. With this growth we have seen the rise of polished distributions, sales-minded distributors, “XX” software is being released, and so free software is gaining commercial success in many fields. Even governments, from Peru to the UK, are now racing to use free software. But governments seem to be the only ones who are talking about switching specifically because they want free software, not just stable, secure and powerful software. It seems to me that many distributors are forgetting the roots of their products, and that's a dangerous thing.

Free software (or, if you like, “open source software”, just replace the terms as you wish) is about more than high quality and low price. The strict definition of both doesn't even mention quality or price, they are merely incidental, and are only potential benefits. The strict definition is that binary programs can be freely copied, that the source code is available to all who want it (and, if the copyright owner desires, those who can pay for it), and that those who have the source code are free to modify it and redistribute the modified versions. Various licenses then dictate exactly how free we are in our use of the source code.

The freedom that these licenses give is the defining factor. And yet if you look at the web sites of many distributors, you won't find any obvious mention of this freedom. Some distributors make passing comments about “the freedom our operating system will give you”, but they rarely explain themselves. If the users, potential or converted, are not fully aware of the defining factor of their operating system, what have they gained? A lot, perhaps, but not the

most important thing of all: the knowledge that their operating system gives them the freedom to use their computer as they wish (within the confines of the law of course!)

Free software country

Now a common argument put forward is that you don't need to know any of that to actually use the operating system. I can use Mozilla without knowing a thing about the license, and not be any worse off in my use thereof. But let me draw a comparison. As citizens of our countries, we are members of the community of citizens known as “society”, and our lives depend on the way that our societies and governments function. Hence at school we learn a little history, and a little about our country, and how it works. But the point is to ensure that we value our nation, we understand our role in our society, and we understand the rights we hold, so that if any of it changes, we can be aware of those changes, and object if we see fit. If enough people object, those changes can be prevented.

Now that is a little idealistic. No state in the world is that democratic or enlightened. The United States began on a good footing, but forgot to educate its children about its constitution, the way it functions, its history, and the world's history. That, combined with other factors I shan't touch on now, has led to a country in which few people know and understand their rights, few know when they are changed, and so few object. A country founded on high and adventurous principles has stagnated and is now run more by corporate interests than the will of an active, educated public. (For the dubious technologist, you need look no further than the DMCA).

The free software movement started on high, adventurous

Fig. 1: Gentoo's social contract makes clear why free software is important



principles. They're enshrined in its licenses, and are well known by most of the programmers who have contributed to free software because they must know the licenses under which they release their work. If programmers were the only people to use free software, you could be fairly sure that the movement wouldn't lose its momentum (though there will always be those who refuse to see how impossible the movement would be without the principles at the root of it). But for users, and even contributors whose work doesn't go under licenses that they'd care to find out about (like documentation and graphics), there's rarely an indication of these principles, let alone an explanation.

How is the movement to remain healthy and principled if few know of or understand its principles? The licenses go

a long way towards protecting users (and, I would argue programmers) from this problem, but it doesn't go all the way. Just as the constitutions and laws of our countries can and have been abused, so the licenses and principles of the free software movement can be abused. And if, over time, what triumphs is "Linux", and not free software, then we have lost.

Governments take a stand

It's interesting to note that some governments have taken a stand for the movement. In countries like Peru and Venezuela, politicians have made the case for free software to be used not only for reasons of stability, cost and secu-

riety, but also for more philosophical reasons to do with the freedom the software offers. Dr. Edgar David Villanueva Nunez, a congressman in Peru, wrote to Microsoft's representative in Peru that

“the foundations of the bill clearly refer to the fundamental guarantees to be preserved and to the stimulus to local technological development. Given that a democratic State must support these principles, it has no other choice than to use software with publicly available source code, and to exchange information only in standard formats.”

While he does also mention security as another key factor, he even points out that “at no point” does the bill refer to “freedom from charges”. The core reason he cites for his decisions to support the bill (which, incidentally, would mandate the use of free software in government, and the exchange of information in open formats) are the principles of the free software movement: the freedom the software elicits and promotes.

What these governments know is that it is the principles of free software which make it so valuable. They promote community, freedom of use, quality, stability and competition. Without them, you'd be left with proprietary software developed by a community with no protection from sharks like Microsoft and Apple. They also know that these principles are not anti-corporate, but that they actually promote software development and help the industry.

Make yourself a Social Contract

Two projects spring to mind that do make some effort to uphold these principles and ensure that their users know about them. They do this through their “Social Contracts”, which define exactly how they will develop their distributions of GNU/Linux, and therefore how their users will benefit from the principles of free software. The first of the two projects, Debian, wrote theirs as the flagpole of their distribution, and they are well known for their principled methodologies. The second project is Gentoo, which took Debian's contract, modified it a little, and posted it on their web site, with a fairly prominent link. Doubtless there are others, so please don't flame me for not mentioning your project. These documents are easy to draft, easy to make prominent, and are an easy way of saying to users: here is how we will empower you; our project will be guided by these principles, and you can be sure to gain these rights. They are not necessarily off-putting, or difficult to understand, and they

do the job, and even link to other resources where users can learn a little more. They also make a show of the principles, citing them as a reason to use free software, which is something that often seems absent in the marketing of many distributions.

The Open Source Initiative was created to end the ambiguity of the term “Free”, and in doing so has shifted the emphasis from freedom to quality. Many now talk about a mythical split between practical and philosophical reasons to use GNU/Linux, despite the fact that one flows from the other. Few try to show businesses the freedoms offered; I tried and failed. Success on this front has come predominantly from persuading hackers and young democratic governments, often in less rich countries.

Conclusion

Perhaps what this tells us is that the principles are accepted by those with more open minds, which have not yet stagnated like the political systems of so many countries, in which established economics, politics and philosophies are doctrine, and questions are uncomfortable. And yet, like a Trojan horse, free software is rolling through the gates of the establishment, ready to jump up and reshape people's ideas about intellectual property and software. This can only happen so long as the foot soldiers know what they're going to be fighting for, and don't just fall asleep, or forget to get into the horse before it goes through the gates.

Copyright information

© 2005 by Tom Chance

This article is made available under the “Attribution-NonCommercial” Creative Commons License 2.0 available from <http://creativecommons.org/licenses/by-nc/2.0/>.

About the author

Tom Chance is a philosophy student, free software advocate and writer. He has worked in various guises with the KDE Project, the Association for Free Software, the Foundation for a Free Information Infrastructure and Creative Commons. You can contact him via his web site, <http://www.tomchance.org.uk> (<http://www.tomchance.org.uk>)

Richard Stallman's blog

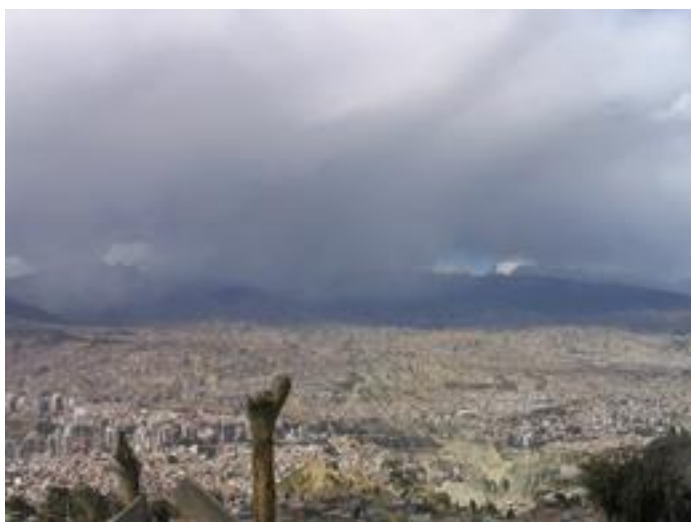
Richard's blog (<http://agia.fsf.org/rms-blog>), from September 2004 to October 2004

Richard Stallman

Bolivia (La Paz) (August 12, 2004 to August 17, 2004)

I am now visiting La Paz, Bolivia. The city is on the edge of the altiplano, starting on the plain at 13000 feet and running down through a connected series of valleys. The result is amazing beauty. Traveling between neighborhoods often means seeing marvelous vistas. The snow-capped mountain Illimani can also be seen from much of the city.

Fig. 1: La Paz in Bolivia



I'm staying in the house of the free software supporter who arranged my speech here. It is on the southern and lower side of the city, in the neighborhood called "Amor de Dios". As an Atheist, I do not like the name very much, but the beauty is amazing. The neighborhood is situated in a valley perhaps 1000 feet across, between a small river and a fairly steep ridge. The ridge runs up from the wall of this house.

Three long streets run parallel to the river. On the other side of the river there is a narrow park and a craggy red cliff. The river was panned for gold hundreds of years ago, and the ridge is said to have some too, but apparently not worth mining.

Today we took a walk up a stairway to a path that goes along the ridge for the length of the neighborhood. The views along the path are marvelous and sooner or later I will get the photos onto stallman.org. We also climbed another stairway to a peak on the ridge, perhaps 50 feet further up, which led to a wood-and-rope bridge that I would not have wanted to cross even if it were in good condition. That stairway was not in good condition either, and about 10 feet of it was so sloping that I was scared to climb down it, scared of falling and once again breaking an elbow or something else. My friends used a large stick to hollow out horizontal places to step, and then with help I was able to climb down. Then they joked that the city ought to pay them for the maintenance work.

The audience for my speech yesterday was disappointingly small; I am told that the students who were supposed to put up posters had a dispute with the director of the computer science department, and went on strike by not putting up the posters. How self-defeating. The other main speaker had been called away on work just a few days before, but he wrote down his speech and it was read for him.

Bolivia is land-locked, so when I said that piracy consists of attacking ships, not of sharing with your neighbor, I said that piracy isn't a problem in Bolivia. Then I remembered that Bolivians still feel strongly about the outlet to the sea that Chile conquered in 1878. So I added, "at present".

Despite the low turnout, the speech has had some good results already. People from the Ministry of Education at-

tended, and have invited me to speak at the ministry on Monday. They are already starting projects to use free software in the schools, and I hope to introduce them to people in Argentina, Brazil and Spain that can give help and advice.

On my next-to-last day in La Paz, I went to see the ancient ruins of Tiwanaku, and Lake Titicaca. My hosts and I hired a taxi for the whole day - it was the only way to go. When we got to Tiwanaku, we took a little too long eating lunch, which forced us to hurry a bit visiting the ruins and the museum.

However, I was glad to have that lunch, because I got to eat a soup with quinoa. Quinoa is a grain that was originally domesticated in the Andes, and I love it, and I had been surprised and a little disappointed not to find any. The reason turns out to be snobbishness: quinoa is considered “what the Indians eat”, so Hispanics generally won’t serve it. (How ironic that in the US you only encounter quinoa in fancy restaurants where the chefs invent new dishes all the time.) My hosts, who are great cooks, reject this snobbishness and often prepare and eat quinoa themselves, but they felt that it wouldn’t be right to make quinoa for a guest like me. So they made rice instead, which isn’t nearly as much fun. However, after I showed them how much I like quinoa, they decided to start. What they made that night, I loved so much that I couldn’t bear to stop eating it.

We reached Lake Titicaca just before nightfall. On the way back, we were at 4000 meters in a thinly populated area, and the stars were so beautiful that I looked at them for many minutes before pulling out my computer to start to answer mail.

On Monday, my last day in La Paz, I gave a brief speech at a university, where hundreds of students had come (no publicity foul-up like the previous time), then a meeting with the Ministry of Education, where the people said they needed to use free software more in the schools, but they could not find people from whom to obtain technical support. Apparently they have no awareness of the hundreds of enthusiastic students who had come to my talks in various cities, showing a vigorous community.

Peru (Lima and Arequipa) (August 17, 2004 to August 22, 2004)

After La Paz, I went to Peru. In Lima I gave three speeches at three universities on three consecutive days, which was rather exhausting since each made a big event of it. Then I went to Arequipa, an inland city to the south of Peru. After

I gave a speech for the GNU/Linux User Group there, they took me to a bullfight.

This was not a Spanish-style bullfight where humans kill a bull. Instead, two bulls fight each other until one of them runs away. Neither the humans nor the bulls get hurt, at least not usually. Although I could see how one can find it exciting, contests don’t continue to fascinate me - and the delays between the matches are long. So after 4 matches I said “let’s go”. (We had to go home to get my things before we could head for the airport.) The photos from Arequipa are now [here](http://stallman.org/photos/peru/arequipa) (<http://stallman.org/photos/peru/arequipa>). The photos from Bolivia are [here](http://stallman.org/photos/bolivia) (<http://stallman.org/photos/bolivia>).

Just before leaving Lima I learned that the free software organization APESOL (“Asociación Peruana de Software Libre”) has set up a web site for people to record that they offer free software support services. If I can put the right people together, maybe something similar can be set up for Bolivia, and this might show the ministry what it needs to see.

Just before leaving Lima I learned that the free software organization APESOL (“Asociación Peruana de Software Libre”) has set up a web site for people to record that they offer free software support services

Home (August 23, 2004)

I have been home now for almost two weeks, which is a long time for me. During this time I’ve set up two sets of 8-foot book-cases, where the front set rotates out to provide access to the back set. This seems to be a solution for the tall space in my new office. On Thursday I am heading for Geneva where consumer organizations are having a meeting about how to deal with WIPO (World Intellectual Property Organization).

Geneva - WIPO (September 10, 2004)

This morning I arrived in Geneva for a meeting of consumer groups on how to deal with the problems caused by WIPO (an organization whose aim is to impose increased “intellectual property rights” on the public). One of my aims at the meeting is to explain why the term [intellectual property rights](http://www.gnu.org) (<http://www.gnu.org>).

org/philosophy/words-to-avoid.html#IntellectualProperty) frames the issue in a way that is harmful to the public, and should be rejected entirely. I have to stay in a hotel this time - something I generally try to avoid, one reason being that hotels in Europe and many other countries participate in a system of surveillance, demanding to see your passport and record information. I wrote "submitted under protest" on the form. The hotel reception agent said, "I'm not the one asking for this; the police insist", and I responded, "Does that make it any better?"

Hotels in Europe and many other countries participate in a system of surveillance, demanding to see your passport and record information. I wrote "submitted under protest" on the form. The hotel reception agent said, "I'm not the one asking for this; the police insist", and I responded, "Does that make it any better?"

The hotel is supposed to have an internet facility, but it has been broken all day. The air conditioning was also not working, and by noon I called reception to report the problem. The agent came up and said, "The air conditioning system here is not very powerful - just wait another hour and the room will get cooler." I went back to work, then took a nap, and when I awoke at 3:30 it was clear the room was, if anything, warmer than before. I complained, and they admitted the system really was malfunctioning. They said that the company that was supposed to repair it was not answering the phone, and they put me in another room which was indeed somewhat cooler.

I turned on the air conditioning, which had been off while the room was unoccupied, and half an hour later I became aware that this room too was getting hotter. The air conditioning system was just heating instead! When I complained again, they admitted the air conditioning system had a central problem. All I could do immediately was turn the ventilation off and hope the room would cool a little.

I cannot sleep when I feel hot unless I am totally exhausted, so I began thinking about leaving; I said I wanted to stay in one of the company's other hotels instead, presuming that not all would have such problems, but they said I could not. I was making plans for how to go about leaving anyway, when the ventilator suddenly seemed to start spontaneously

Fig. 2: Oslo



to blow some cool air. I thought it was working - though now I am not sure - turning it up to maximum strength has not increased the cooling. I have the feeling that the staff have been manipulative, and less than truthful, at every stage of this. The hotel is part of the Manotel group, in case you're looking for hotels in Geneva not to stay in.

Norway (September 14, 2004 to September 19, 2004)

I went to Norway to speak at a Java conference, and I was probably the only person in the room who did not know the Java language. (I expect to be in Java a month from now, and I've been studying Indonesian on and off for a couple of years, but I have never learned to read Java.)

I went to Norway to speak at a Java conference, and I was probably the only person in the room who did not know the Java language

I gave a quick explanation of free software and then explained about the **Java Trap** (<http://www.gnu.org/philosophy/java-trap.html>).

After that, and a speech at a university the next day, I went by car to the small town of Skei (pronounced somewhere between "shy" and "shay"), in the west of Norway where the fjords are. It is in the middle of mountains, some of which have glaciers. This was the first time I had been anywhere in Norway aside from Oslo. (I am guessing that the Oslo area became so populous and important because it is

the main flat part of the country.) On the way, and there, I took a lot of photos (<http://www.stallman.org/photos/norway>).

The glaciers are already considerably smaller than they once were. Go see them now, before global warming melts them. In Skei there were also activists for computerized community currencies. They explained about how their system would work.

Luxemburg (September 27, 2004 to September 28, 2004)

Yesterday I visited Luxemburg for the first time. Now I have been in all the countries of the European Union.

My speech yesterday was something I rarely do: a debate. The first speaker was a patent lawyer. The organizers said it would be easier to set up the event if they could invite him too, and this person wasn't a cunning orator, so I took the risk - and I wiped the floor with him.

His speech presented fine examples of all the common confusions that I like to explain in my speeches about software patents. Not that he himself was confused - he was only trying to lead the audience astray. For instance he referred to "patenting software", which implies that software idea patents cover entire programs. He also described software idea patents as a way to "protect software", from which one would never guess that the main effect of software patents on software developers is to put them at risk of being sued. (See [this link \(http://softwarepatents.co.uk/\)](http://softwarepatents.co.uk/) for more explanation.)

I'm told that an assistant to the relevant minister was there. Perhaps the speech will do some good. We are trying to ask various EU countries to change their votes on the issue; just a couple more small countries will be enough to win the battle.

This morning I woke up for no particular reason before 8am and could not get back to sleep. So I was willing to travel to the University of Luxemburg, where a newspaper interview was supposed to occur. Reportedly a control-freak PR person at the university had decided to make both me and the reporter go there, even though it would have been more convenient for both of us if the reporter had 'come to the place I was staying.

After that interview I took a few trains, and now I'm in Essen, Germany. After my speech here I have to take more trains to Amsterdam this evening. "Essen" means "eating", so it's delightfully ironic that my visit here is so short that I won't have time to eat.

Two hours from death? (September 30, 2004)

A week ago my plan was to give two speeches in Amsterdam on Wednesday Sep 29, then go to Paris on Sep 30. But in Geneva I learned that there was an e-Democracy conference in Paris on Sep 30 at which it would be useful for me to speak. Francis Muguet was organizing my participation, but it turned out on Tuesday that the only time I could speak was the morning.

So various people began trying to find a way I could get there early enough to do this. The last flight in the evening was too early, we discovered on Wednesday morning. There was a train leaving at 2020 which I could have taken if I ran out of the speech a little early and canceled my invitation to dinner. I was thinking of doing that when someone had the idea that people in the free software community could drive me to Paris. Ultimately we chose that solution. Three free software enthusiasts met me after dinner, borrowing my host's car.

Departure was scheduled for 10pm, but was delayed because my halo was missing. It had fallen out of its bag while that was in the back of a car, and rolled under a seat, where we did not see it. After looking in the other possible places such as the room where I had spoken, and not finding it, we searched the car thoroughly.

We should have dropped me off in Paris around 330pm, but we got lost there. They were following a navigation system in the car, and it got confused. When I recognized where we were headed and gave directions, there was a misunderstanding that got us lost again. Eventually we ended up at the Etoile, and the navigation system started working. We got to Francis Muguet's apartment and they dropped me off. The three people from Amsterdam headed back, but did not arrive. They had an accident.

Despite the many things I had to do, I got about 3 hours of sleep before I had to go and speak. I gave a good short speech to a workshop (perhaps a third of the conference), and then Francis for me to do more. The French Minister of Industry was scheduled to speak that afternoon; he is the one who decides the French policy on software patents. It was arranged that I would be able to ask him a question for certain, if he accepted any.

While walking into the conference room, and discussing with Francis what I should say in my question, I received a phone call telling me that the car had crashed and one of the men who had driven with me was dead.

This was a sobering thought. I did not feel personal grief, because it was not a personal loss. The three were strangers who had helped me for the sake of free software, rather than

personal friends, and we only barely had begun to be acquainted. However, it was weighty to realize that someone had died because he had helped me get to Paris for this meeting. I did not feel guilt about his death - I did not cause the accident - but I felt a responsibility to make his death count for something.

I asked the minister whether France would sustain the European Parliament's vote against software patents. His answer showed total incomprehension; he spoke about the virtue of copyright and the "principle" of "intellectual property" (thus illustrating why people must reject the use of this term). I felt a sense of total failure. Francis told me he cried at this point.

As the minister was leaving, I had a chance to exchange a couple of sentences with him. He really did not know how patents affect software developers. Francis says that the minister wanted to talk with me further about the issue. I am on my way out of France right now, and may not have a chance to be back in Paris until it is too late. But maybe we can find someone else who can follow up on this contact.

It was only later, when I saw there had been some public discussion of whether I was in the car at the time of the accident, that I realized that I too had had a somewhat narrow escape. If the accident had happened two hours earlier, I would have been in it.

Australia (October 3, 2004 to October 18, 2004)

I was invited to Australia so as to speak at the Builder conference, which was canceled shortly before I got there (but they had already bought my tickets). This did not mean the visit was wasted, since I had arranged 9 other speeches. The Australian Senate had attached some conditions to the US-Australia Free Trade Agreement, and it looked like the US might reject the treaty as a result, which would give Australia a second chance to escape. I arranged to give several speeches about the danger of software patents.

 The Australian Senate had attached
 some conditions to the US-Australia
 Free Trade Agreement, and it looked like
 the US might reject the treaty as a result,
 which would give Australia a second
 chance to escape

During the first week there I was contacted by someone who knows the parents of Hans Bakker, who died in the accident

Fig. 3: In Australia



returning from Paris. I got their address and sent them a message of condolence, which was not easy to write. Although the minister said he would meet with me, it seems this won't happen - there is only one occasion I could arrange to be in Paris between now and the vote scheduled for a month from now, and he can't make it then.

Half-way through my visit, Australia held a general election. The conservative "Liberal" party, which supports Bush and the treaty, gained support after a campaign based on lies. Howard lied to them about the war in Iraq, too.

Their previous electoral campaign, three years ago, was based on lies that boat people were throwing their own children into the water to force a rescue. After the election, navy personnel testified this was because their boat was in the process of sinking. So I was not really surprised when, a few days after the election, I heard on the radio that their optimistic economic projections were exaggerated and would not come to pass. Just goes to show, if you tolerate a government that lies about minor things like human rights and refugees and war, soon they will start lying about your money too.

The election outcome could give them control of the Senate, which could mean that Australia wastes its second chance and approves the treaty.

I spent much of election day going to visit a lorikeet named Scratchy, who I had had a wonderful time with on my previous visit several years ago. However, Scratchy was not in a friendly mood this time - he was in love with his bell, and didn't really want to play with anyone, and tended to nip at people.

Just before leaving Australia, I visited a couple of cockatiels that sat on my hand and shoulder and chattered. I tried to

teach them to say “Are you a bird?”, but it must take more time than that.

Malaysia (October 19, 2004 to October 21, 2004)

I spent two days in Malaysia, where I had my first chance to try conversing in Malay (it is pretty similar to Indonesian, which I have been studying), and a chance to try the particular food tradition of people of mixed Malay and Chinese descent. My host said it was the only one likely not to be too spicy for me. However, one of the dishes that the waiter said was “not spicy at all” turned out to be too spicy for me to eat.

The next day was my speech, which went well. In the evening I visited the twin towers of Kuala Lumpur, which were beautiful. In the photos I took, they appear to curve towards each other - I think that is due to distortion in the wide-angle lense that I needed in order to get the whole of them into one photo.

Jakarta (October 21, 2004 to October 22, 2004)

The following day I managed to converse a little with the taxi driver on the way to the train to the airport. When I arrived in Jakarta, I was surprised to see a man with a sign with my name on it waiting at the exit from the jetway. It turned out he had been sent there to help me get through immigration and customs easily. Everything went completely smoothly with his help, and I was able to converse with him too. Also with my hosts that were with me in the car coming back, and at lunch. Most of them didn’t eat, they just watched, as it is Ramadan. I took the opportunity to explain to them in Indonesian that MacDonalds’ “fast food” is meant for helping people fast - not for eating.

It was quite a pleasure to feel that I can now speak a fourth language. However, it is a constant effort and I can only do it when I am feeling very awake. This morning I am finding it hard to handle sentences that yesterday I could handle easily.

We went to an outdoor dinner at the university where I am speaking, with many of the students involved in free software there. Bats were flying around just above our heads, as we had an appetizer which I first thought was made of broad and thick mushrooms soaking in coconut sauce. But they were not mushrooms, they were a sort of pancake that only looks like a mushroom to me. I sang the free software song.

Then a musical group began playing and singing in a style called campur sari, which is a fusion of western and Javanese music that I gather is rather popular; but it is too much western pop for my tastes. For a while they stopped and some girls performed a dance in a style from Aceh, mostly sitting down either each separately or in a line, involving a lot of clapping and moving in different synchronized groups that move through each other. That was interesting.

Then a singer came out and began singing “You’re just too good to be true”. I don’t like American popular music terribly much, so I decided to flirt/tease by catching her eye and pretending I thought she was talking about me.

I was surprised by the response: she motioned for me to come and sing with her. (I should not have been surprised, because I’ve read about that custom.) Partly I tried to sing along with her, to the extent I remember that song (I’d never wanted to sing it), and partly in humoristic response, saying “You’ll learn more about me soon”.

Then she asked me to dance along with her, so I improvised a dance, combining my Balkan folk dance experience and what I’ve seen of Indonesian dancing. It was a big hit. But after about three minutes I was worn out and had to sit down. At that point I was a bit too tired to figure out how to explain this to her in Indonesian (she wanted me to continue). I had to say it in English, and then I felt disappointed with myself. She invited a few others to dance. Later several of us danced together. It was a lot of fun. I chatted with her (in English) for a while after her performance was done, as I waited for people to try to solve a network problem that prevented me from doing ssh to the GNU servers. The problem was impossible to solve, so I had to go to another building to do that mail transfer.

Copyright information

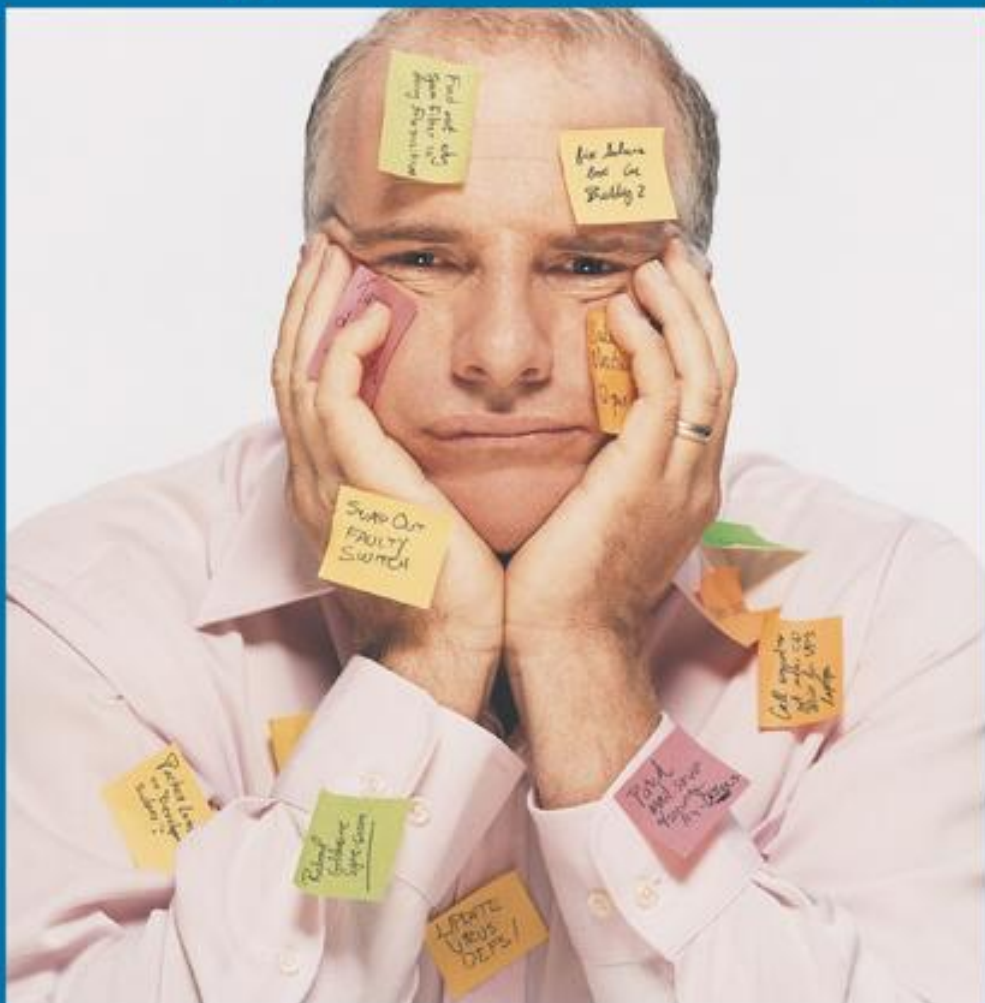
© 2005 by Richard Stallman

Verbatim copying and distribution of this entire article is permitted in any medium without royalty provided this notice is preserved.

About the author

Richard Stallman is the founder of the GNU Project, launched in 1984 to develop the free software operating system GNU. The name “GNU” is a recursive acronym for “GNU’s Not Unix”.

Losing track of things?



There is a better way.

RT.

Enterprise-grade issue tracking.
Open source customizability.
Commercial support.

Download it now!

Best Practical Solutions offers training, support and custom development services. Check out our website or contact us for details.

www.bestpractical.com
sales@bestpractical.com

»|« BEST
PRACTICAL™



A PLANT NEEDS WATER TO GROW

**Free Software
MAGAZINE**

BY SUBSCRIBING YOU WILL BE SUPPORTING A MAGAZINE WHICH BELIEVES IN FREE SOFTWARE. ALL OUR ARTICLES ARE RELEASED UNDER THE GNU FREE DOCUMENTATION LICENSE, ENHANCING EXISTING INFORMATION ON FREE SOFTWARE.

SUBSCRIBE NOW!

[HTTP://WWW.FREESOFTWAREMAGAZINE.COM/SUBSCRIBE](http://WWW.FREESOFTWAREMAGAZINE.COM/SUBSCRIBE)

THE OPEN VOICE NEEDS SUBSCRIBERS TO PROSPER!

